P.1847/93

4 1993

informatyka

WYDAWNICTWO SIGMA NOT

Cena 25 000 zł

informatyka

nr 4 1993
Kwiecień
Rok wydania XXVIII

Miesięcznik
ISSN 0542-9951
INDEKS 36124

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
mgr Teresa JABŁOŃSKA
(sekretarz redakcji)
Władysław KLEPACZ
(redaktor naczelny)
mgr inż. Jan RYŻKO
dr Zdzisław SZYJEWSKI

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek
Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych
redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu 00-950 Warszawa ul. Mazowiecka 12 telefon: 27-43-66 telefaks: 19-21-87 teleks: 814877

W numerze:

	Strona
Programowanie systemów transputerowych – *Tomasz Kalinowski*	1
Systemy zarządzania relacyjnymi bazami danych oraz ich przydatność do budowy informatycznych systemów zarządzania – *Lucyna Korbel, Elżbieta Miłosz, Marek Miłosz*	8
Klasyfikacja narzędzi wspomagania CASE – *Zdzisław Szyjewski*	13
Systemy wspomagania decyzji z bazą wiedzy – *Henryk Sroka*	16
Model konwersacyjny CAL w środowisku KnowledgePro – *Jacek Kotula, Jarosław Zyguła*	21
Prawne środki ochrony programów komputerowych – możliwe kierunki zmian w USA – *Andrzej Matlak*	25

W najbliższych numerach:

- **Danuta Bartoszek charakteryzuje ECHO – komputerowy serwis informacyjny Europejskiej Wspólnoty Gospodarczej.**
- **Krzysztof Zieliński i Grzegorz Czajkowski prezentują środowisko do przetwarzania równoległego i rozproszonego w sieciach stacji roboczych – LINDA.**
- **Dariusz Gaik i Andrzej Przybylski opisują pakiety graficzne współpracujące z oprogramowaniem narzędziowym APW.**
- **Jan Goliński zajmuje się problemami projektowania systemów informatycznych.**
- **Andrzej Matlak opisuje prawne środki ochrony programów komputerowych w USA.**

Warunki prenumeraty na 1993 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573

Wpłaty na prenumeratę od marca br. przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 25 000 zł, ulgowa 18 750 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 75 000, półroczna 150 000, roczna 300 000
Ulgowa: kwartalna 56 250, półroczna 112 500, roczna 225 000.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 178/93

P.1877/93
POLITECHNIKA ŚLĄSKA Biblioteka Główna w Gliwicach

TOMASZ KALINOWSKI
Instytut Podstaw Informatyki
Polska Akademia Nauk
Warszawa

Programowanie systemów transputerowych

Nazwa transputer obejmuje rodzinę mikroprocesorów firmy IMMOS, które charakteryzują się wysoką wydajnością oraz rozbudowanymi mechanizmami współbieżności i komunikacji. Transputer stanowi w istocie sprzętową realizację modelu CSP [3], w którym statycznie definiowane procesy komunikują się synchronicznie przez jednokierunkowe kanały danych. Mechanizmy sprzętowe transputera wspomagają w jednolity sposób wykonywanie zbioru procesów współbieżnych zarówno na pojedynczym procesorze, jak i na ich sieci. Dzięki temu na bazie transputera można szczególnie łatwo budować skalowalne systemy wieloprocesorowe.

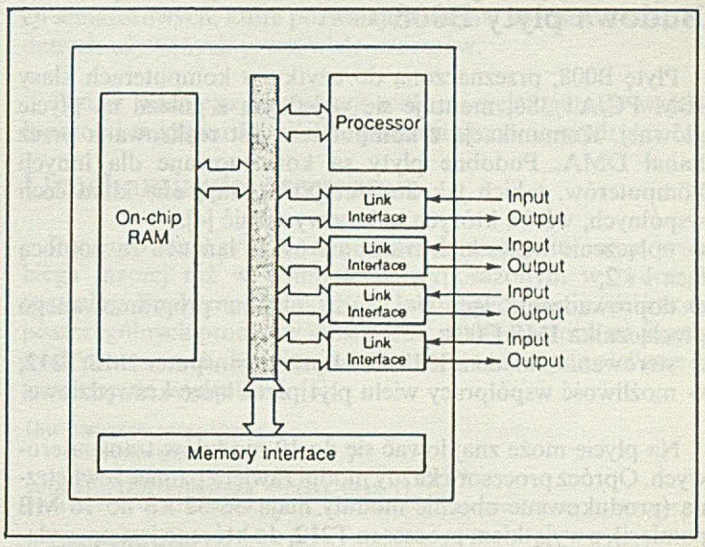
Mimo swojej prostoty i efektywności, komunikacja między węzłami sieci transputerowej podlega pewnym ograniczeniom, do których należy m.in. wymóg istnienia fizycznego połączenia między współpracującymi procesorami. Jest oczywiste, że w wielu zastosowaniach spotykamy się z koniecznością współpracy procesorów, które działają w węzłach sieci nie będących bezpośrednimi sąsiadami. Celem autora artykułu było stworzenie mechanizmu, który usunąłby wspomnianą niedogodność.

Celem lepszego zrozumienia tematu należy na wstępie przypomnieć zasadę działania transputera, opisywaną już w INFORMATYCE [2], oraz przedstawić budowę najczęściej stosowanej w komputerach klasy PC płyty wieloprocesorowej B008, a także używany język Parallel C. Z tej również przyczyny zostanie omówiony schemat uruchamiania programów, odmienny niż w systemach wyposażonych w centralny procesor. Podstawową treścią artykułu jest opis zrealizowanego modułu komunikacyjnego, który zapewnia jednolity sposób komunikacji między dowolnymi węzłami sieci transputerowej.

Architektura transputerów

Transputery mają stosunkowo prosty zestaw instrukcji. Podobnie jak dla procesorów typu RISC, większość operacji wykonywana jest w jednym cyklu zegara. Dłużej są realizowane instrukcje przełączania procesów oraz instrukcje realizujące czynności specyficzne dla danego typu procesora, jak np. przetwarzanie grafiki. Do wspólnych rozwiązań zastosowanych we wszystkich typach transputerów można zaliczyć [4]:

- procesor całkowitoliczbowy,
- szybką pamięć statyczną,
- szeregowe połączenie komunikacyjne z innymi transputerami,
- wewnętrzny zegar,
- interfejs pamięci zewnętrznej.



Rys. 1. Architektura transputera

Strukturę procesora przedstawiono na rysunku 1 [5].

Procesor nie został wyposażony w pamięć podręczną (ang. cache), w której byłyby przechowywane kolejne rozkazy lub najczęściej używane dane, dlatego też programista musi podjąć decyzję, które fragmenty kodu zostaną umieszczone w wewnętrznej pamięci statycznej.

Wewnątrz procesora może działać współbieżnie praktycznie nieograniczona liczba procesów. Procesor zawiera moduł szeregujący (ang. *scheduler*), który zarządza dwiema klasami procesów. Procesy wysokopriorytetowe są wykonywane aż do zakończenia lub są zawieszane w oczekiwaniu na operację wejścia-wyjścia. Procesy niskopriorytetowe dzielą czas procesora, mogą być więc dodatkowo wywłaszczone po upływie kwantu czasu.

Procesy wykonywane na transputerach są synchronizowane na zasadzie przekazywania komunikatów. Z punktu widzenia programisty niemalże nie ma różnicy, czy wiadomość jest przesyłana między procesami wykonywanymi na tych samych, czy też różnych procesorach. Przesłanie informacji między wykonywanymi na tym samym procesorze procesami polega na kopiowaniu danych między ich obszarami roboczymi. Komunikacja jest synchronizowana za pomocą słowa sterującego kanału, umieszczonego pod dowolnym adresem w pamięci. Transmisja szeregowa między procesorami jest realizowana przez układ DMA. Poprawność przesłania nie jest sprawdzana.

Jak dotąd firma INMOS wyprodukowała trzy rodziny transputerów:

- T2 – procesory 16-bitowe,

- T4 – procesory 32-bitowe,
- T8 – procesory 32-bitowe, wyposażone w jednostkę zmiennoprzecinkową (1,3 MFLOPS).

Od dość dawna zapowiadany jest nowy procesor T9000 [9], binarnie zgodny z dotychczas produkowanymi, ale o blisko dziesięciokrotnie większej mocy obliczeniowej (13,8 MFLOPS). Nowy układ ma 16 kbajtów pamięci cache oraz dodano w nim instrukcje realizujące operacje semaforowe. Znacznie rozbudowano możliwości komunikacyjne. Wiadomości dzielone są na pakiety, opatrzone nagłówkiem. Układ obsługi kanałów wirtualnych pozwala na umieszczenie wielu kanałów w jednym fizycznym łączu. Niestety pojawienie się tego układu na rynku jest ciągle odkładane.

Budowa płyty B008

Płytę B008, przeznaczoną do użytku w komputerach klasy IBM PC/AT/386, montuje się w jednym z gniazd na płycie głównej. Komunikacja z komputerem jest realizowana przez kanał DMA. Podobne płyty są konstruowane dla innych komputerów, takich jak Sun lub Vax. Mają one kilka cech wspólnych, wśród których należy wymienić [4]:

- połączenie wszystkich transputerów w łańcuch za pomocą łącz 1 i 2;
- doprowadzenie łącz 0 i 3 do układu programowanego przełącznika IMS C004;
- sterowanie układem IMS C004 przez transputer IMS T212;
- możliwość współpracy wielu płyt przez łącze krawędziowe.

Na płycie może znajdować się do 10 modułów transputerowych. Oprócz procesora każdy moduł zawiera pamięć zewnętrzną (produkowane obecnie moduły mają od 32 kB do 16 MB pamięci), z wyjątkiem procesora T212, do którego jest przesyłany program inicjujący pracę przełącznika C004, który wyposażono jedynie w 2 kB pamięci wewnętrznej. Tylko pierwszy procesor na płycie może wymieniać informację z jednostką centralną. Użytkownik ma możliwość programowego połączenia dowolnego łącza 0 do łącza 3 dowolnego modułu. Schemat płyty przedstawiono na rysunku 2. [8].

Z płyt B008 mogą być budowane większe systemy. Ostatni procesor na płycie jest wówczas łączony z pierwszym procesorem kolejnej płyty (*PipeHead*/*PipeTail*), której połączenie z komputerem PC zostaje przerwane. Oddzielny wektor tworzą procesory T222 (*ConfigUp*/*ConfigDown*). Układ C004 może obsłużyć 32 łącza. Jest to więcej niż liczba łącz doprowadzonych od modułów znajdujących się na płycie, dlatego też część linii (*EdgeLink*) została wyprowadzona na łącze krawędziowe, co pozwala na komunikację między procesorami znajdującymi się na różnych płytach.

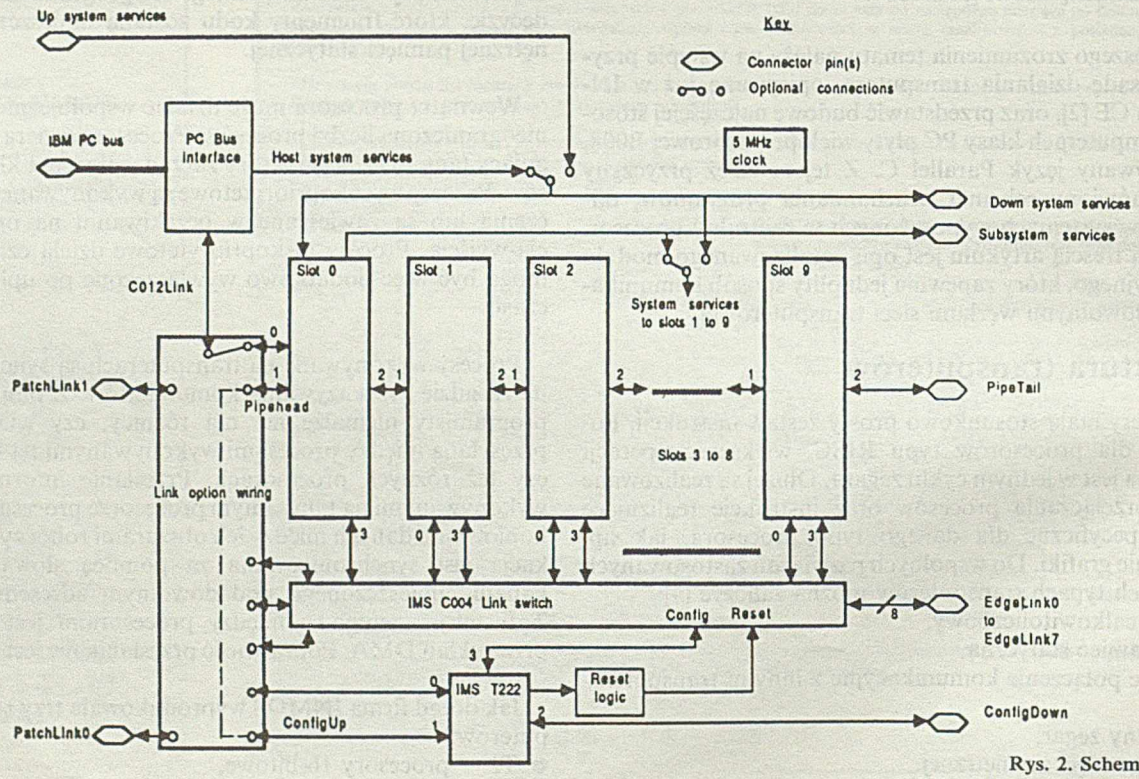
W systemie transputerowym można wyodrębnić podsystemy. Płyta może nadzorować (tzn. inicjować pracę procesorów, zatrzymywać wykonanie w celu zanalizowania stanu procesora, obsługiwać zgłoszone błędy) działania podsystemu złożonego z innych płyt i dołączonego do portu *Subsystem*. Płyty tworzące podsystem są łączone przez porty *Up* oraz *Down*.

Język Parallel C

Pierwszym językiem wysokiego poziomu, w którym programowano transputery był Occam. Projekt tego języka powstał równolegle z projektem procesora. Zgodnie z modelem zaproponowanym przez Hoare'a [3], program w Occamie składa się ze współbieżnie wykonywanych procesów przesyłających między sobą komunikaty. Occam dokładnie opisano w INFORMATYCE [1]. Zbiór dostępnych w tym języku typów danych oraz metod ich strukturalizacji jest dość ubogi. Dlatego też wśród użytkowników systemów transputerowych znaczną popularność zdobywają, wzorowane na modelu CSP, rozszerzenia powszechnie stosowanych języków programowania, takich jak C, Pascal, a nawet Fortran.

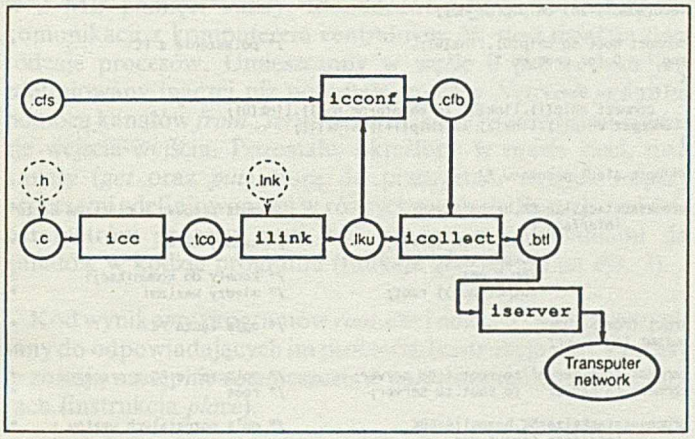
Implementacji modułu komunikacyjnego autor dokonał w języku Parallel C [5, 6], ponieważ podobnie jak C jest on wygodnym narzędziem do tworzenia oprogramowania systemowego.

Mechanizmy współbieżności Parallel C zostały zrealizowane w postaci wywołań bibliotecznych. Składnia języka nie została



Rys. 2. Schemat płyty B008

zmieniona i jest zgodna z normą ANSI C. W plikach *process.h* i *channel.h* zdefiniowano typy *Process, Channel* oraz *Semaphore*. Procesami stają się funkcje, których pierwszym parametrem jest wskazanie deskryptora procesu. Uruchamianie procesu (*ProcPar*) wymaga uprzedniej alokacji (*ProcAlloc*), w czasie której należy podać nazwę funkcji tworzącej proces, rozmiar stosu, liczbę parametrów funkcji oraz parametry wywołania. Program przedstawiony na rysunku 3 alokuje w każdym węźle dwa procesy: *rec*, który przyjmuje wiadomość nadesłaną z poprzedniego węzła, oraz *send*, który wysyła tę wiadomość do kolejnego procesora. Uruchomienie tej pary procesów poprzedza przyporządkowanie używanych w programie kanałów (*inp_chan* i *out–chan*) do zdefiniowanych w opisie konfiguracji połączeń między procesorami (funkcja *get_param*) oraz alokacja kanału służącego do komunikacji wewnątrz węzła (*move_msg*).



Rys. 3. Schemat uruchamiania programu

```
#include <process.h>
#include <channel.h>
#include <stdlib.h>
#include <misc.h>

void get_num(Process *p,Channel *from,Channel *move)
{
  int num;
  num = ChanInInt(from);
  ChanOutInt(move,num);
}

void send_num(Process *p,Channel *move,Channel *to)
{
  int num;
  num = ChanInInt(move);
  ChanOutInt(to,num);
}

main()
{
  Channel *inp_chan,*out_chan;
  Channel *move_msg;
  Process *rec,*send;

  if ((inp_chan = get_param(1)) == NULL)
    exit_terminate(EXIT_FAILURE);
  if ((out_chan = get_param(2)) == NULL)
    exit_terminate(EXIT_FAILURE);
  if ((move_msg = ChanAlloc()) == NULL)
    exit_terminate(EXIT_FAILURE);

  if ((rec = ProcAlloc(get_num,0,2,inp_chan,move_msg)) == NULL)
    exit_terminate(EXIT_FAILURE);

  if ((send = ProcAlloc(send_num,0,2,move_msg,out_chan)) == NULL)
    exit_terminate(EXIT_FAILURE);

  ProcPar(rec,send,NULL);

  exit_terminate(EXIT_SUCCESS);
}
```

Wydruk 1. Przykładowa realizacja procesów współbieżnych w języku Parallel C

Procesy współbieżne dzielą się na: synchroniczne (*ProcPar*) – wykonanie procesu macierzystego zostaje zawieszone do czasu zakończenia wykonania wszystkich procesów potomnych lub asynchroniczne (*ProcRun*) – wykonywane współbieżnie z procesem wywołującym. Możliwe jest nadanie procesom priorytetu (*ProcRunHigh, ProcRunLow*), a także uruchomienie pary procesów: wysoko- oraz niskopriorytetowego (*ProcPriPar*).

Komunikacja między procesami może odbywać się przez kanały i zmienne globalne. W Occamie zmienne współdzielone mogły być jedynie odczytywane, a kompilator sprawdzał spełnienie tego warunku. W kanale można przesyłać znaki, liczby całkowite lub bufory danych o rozmiarze ustalonym przez nadawcę i odbiorcę.

W Parallel C zawarto konstrukcje umożliwiające oczekiwanie na informacje z wielu źródeł. Funkcje sprawdzające stan dozoru udostępniają numer kanału, do którego nadesłano wiadomość. Oczekiwanie może być bezterminowe (*ProcAlt*) lub ograniczone w czasie (*ProcTimerAlt*). Możliwe jest też jedynie sprawdzanie bieżącego stanu kanału (*ProcSkipAlt*).

Istotnym odstępstwem od modelu CSP jest dołączenie operacji semaforowych, które pozwalają znacznie uprościć dostęp do danych używanych przez wiele procesów.

Wykaz ważniejszych funkcji realizujących operacje współbieżności w języku Parallel C przedstawiono na wydruku 1.

Uruchamianie programu

Przygotowanie programu w systemie transputerowym przebiega inaczej niż w komputerze wyposażonym w procesor centralny. Po skompilowaniu (*icc*) i skonsolidowaniu (*ilink*) kod poszczególnych procesów musi zostać rozmieszczony na fizycznych procesorach. Kolejność wykonania tych czynności przedstawiono na wydruku 2. [5].

Uruchamianie procesów:

synchronicznych:

```
void ProcPar(Process *deskr_proc, ..., NULL);
void ProParList(Process **lista_deskr);
void ProcPriPar(Process *proc_wysokoprior, Process *proc_niskoprior);
```

asynchronicznych:

```
void ProcRun(Process *deskr_proc);
void ProcRunHigh(Process *deskr_proc);
void ProcRunLow(Process *deskr_proc);
```

Przesyłanie komunikatów:

wysyłanie wiadomości:

```
void ChanOut(Channel *kanal, void *bufor_danych, int rozmiar_bufora);
void ChanOutInt(Channel *kanal,int wysyl_liczba);
void ChanOutChar(Channel *kanal,char wysyl_znak);
```

odbiór wiadomości:

```
void ChanIn(Channel *kanal, void *bufor_danych, int rozmiar_bufora);
int  ChanInInt(Channel *kanal);
char ChanInChar(Channel *kanal);
```

Oczekiwanie na nadejście wiadomości z wielu źródeł:

oczekiwanie bezterminowe:

```
int ProcAlt(Channel *kanal, ...,NULL);
int ProcAltList(Channel **lista_kanalow);
```

sprawdzenie stanu kanałów:

```
int ProcSkipAlt(Channel *kanal, ...,NULL);
int ProcSkipAltList(Channel **lista_kanalow);
```

oczekiwanie ograniczone w czasie:

```
int ProcTimeAlt(int czas_oczekiwania,Channel *kanal,...,NULL);
int ProcTimeAltList(int czas_oczekiwania,Channel **list_kanalow);
```

Obsługa semaforów:

```
Semaphore *SemAlloc(int wartosc_pocz);
SemSignal(Semaphore *deskr_sem);
SemWait(Semaphore *deskr_sem);
```

Wydruk 2. Mechanizmy współbieżności w Parallel C

Na podstawie zawartego w pliku konfiguracyjnym (*.cfs*) opisu procesorów i ich wzajemnych połączeń, program konfigurujący *icconf* sprawdza, czy wygenerowany kod może być wkonany na danym typie transputera, rozmieszcza procesy i kanały komunikacyjne na wybranych procesorach i łączach, a także określa wielkość pamięci, jaka musi być przydzielona dla wykonania procesu na danym procesorze. Dane o rozmieszczeniu procesów i skonsolidowany program są następnie prze-

twarzane przez program *icollect* na program ładowalny, który dodatkowo zawiera sekwencje inicjacyjne dla procesorów oraz program rozsyłający kod do wszystkich węzłów sieci. Ostatnim etapem jest załadowanie kodu programu do systemu wynikowego przez program *iserver*.

Uruchamianie programu współbieżnego jest z natury znacznie trudniejsze niż programu sekwencyjnego. Dlatego też ogromną rolę odgrywają narzędzia, które mogłyby wspomagać ten proces. *Idebug* firmy INMOS umożliwia śledzenie wykonania programów w sieci transputerowej na poziomie kodu źródłowego oraz badanie stanu procesora (zawartość pamięci i rejestrów, kolejek, procesów, stanu łącz). Program pozwala zatrzymać pracę pojedynczego procesora lub przerwać wykonanie programu oraz zbadać stan wszystkich procesorów w trybie post-mortem. Jak w większości tego typu programów istnieje możliwość ustawiania pułapek, odczytania i zmiany wartości zmiennych. *Idebug* umożliwia sprawdzenie kanału. Jeśli istnieje proces oczekujący na przesłanie wiadomości przez ten kanał, to program pozwala na zbadanie stanu tego procesu. Niestety, użycie debuggera wymaga zmiany konfiguracji uruchamianego programu. *Idebug* rezerwuje dla swoich potrzeb pierwszy procesor, co powoduje zmiany w przyporządkowaniu łącz (np. komunikacja ze „światem zewnętrznym" jest realizowana nie przez łącze 0, ale przez łącze 1 następnego procesora).

Konfigurowanie programów transputerowych

Konfiguracja płyty B008 (a właściwie sposób połączenia linii 0 i 3 każdego procesora, gdyż połączenie transputerów w łańcuch nie podlega modyfikacjom) jest realizowana przez program MMS (*Module Motherboard Software*). Program ten, na podstawie listy połączeń stałych, programowych oraz linii wyprowadzonych na łącze krawędziowe, przygotowuje sekwencję rozkazów dla przełącznika C004. Konfiguracja jest zapamiętywana do chwili wyłączenia zasilania lub zapisania nowej konfiguracji.

Topologia procesorów w programie użytkownika musi być zgodna z konfiguracją sprzętu. Opis połączeń, przeznaczony dla programu *icconf* jest dokonywany za pomocą języka o składni wzorowanej na języku C:

- budowa wyrażeń jest taka, jak w języku C;
- identyfikatory mają zasięg globalny;
- są dostępne instrukcje: pętli *rep* oraz warunkowa *if*;
- jest dozwolone deklarowanie tablic procesorów, procesów, kanałów, a także tablic dowolnego typu zdefiniowanego przez użytkownika;
- instrukcja *connect* służy do deklarowania połączeń między procesorami i kanałami zadeklarowanymi na różnych procesorach.

Opis konfiguracji składa się z opisu sieci procesorów, sieci procesów oraz odwzorowania procesów i kanałów na poszczególne procesory i łącza.

Sieć sprzętowa określa typ, rozmiar pamięci i liczbę łączy każdego procesora oraz połączenia między procesorami. W pliku *setconfig.inc*, automatycznie dołączanym do każdego pliku konfiguracyjnego, umieszczono definicje typów wszystkich procesorów produkowanych przez firmę INMOS.

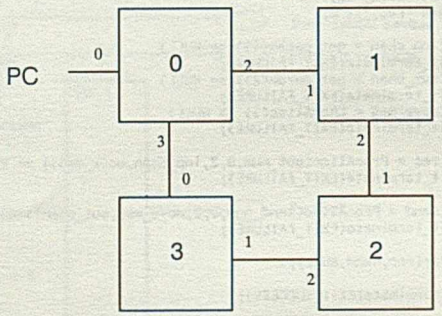
W skład opisu **sieci procesów** wchodzą deklaracje procesów oraz łączących je kanałów. Proces jest opisywany przez następujące atrybuty:
- rozmiar stosu;
- rozmiar sterty;
- interfejs procesu: kanały wejścia-wyjścia oraz parametry przekazywane do procesu;
- priorytet;
- rozmieszczenie w pamięci segmentów kodu, danych, stosu i sterty.

Ostatnim etapem konfiguracji jest przyporządkowanie plików zawierających kod wynikowy odpowiadającym im procesom oraz przypisanie procesów do właściwych procesorów.

Rozmieszczenie kanałów na łączach może być dokonane automatycznie lub zdefiniowane przez użytkownika.

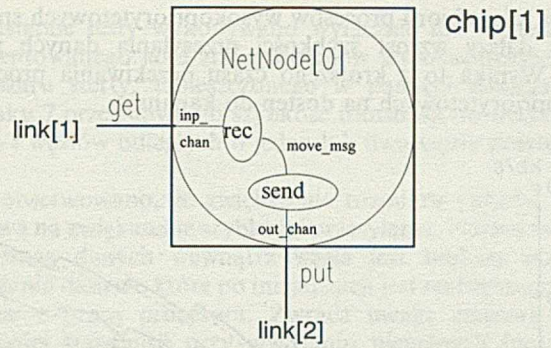
```
val proc_no 4;

/*opis sieci transputerowej */

T800(memory=2M) chip[proc_no];

connect host to chip[0].link[0];                    /* polaczenie z PC              */
rep i = 0 for proc_no-1
{
  if ( i == 0)
    connect chip[i].link[3] to chip[proc_no-1].link[0];
  connect chip[i].link[2] to chip[i+1].link[1];
}

/* opis sieci procesow */

process(stacksize=8K,heapsize=40K,                  /* opis procesu w pierszym wezle*/
        interface (
                  input from_server,                /* kanaly we/wy                 */
                  output to_server,
                  input get,                        /* kanaly do komunikacji        */
                  output put)) root;                /* miedzy wezlami               */

input from_server;                                  /* opis lacza PC                */
output to_server;

connect from_server to root.from_server;            /* polaczenie PC z procesem     */
connect to_server   to root.to_server;              /* root                         */

process(stacksize=8K,heapsize=40K,                  /* opis pozostalych wezlow      */
        interface (input get,
                   output put)) NetNode[proc_no-1];


connect root.put to NetNode[0].get;                 /* polaczenia miedzy procesami  */
connect root.get to NetNode[proc_no-2].put;
rep i = 0 for proc_no-2
  connect NetNode[i].put to NetNode[i+1].get;


/* rozmieszczenie procesow */

use "root.c8s" for root;                            /* przypisanie kodu do procesow */
rep i = 0 for proc_no -1
  use "node.c8s" for NetNode[i];

place root on chip[0];                              /* rozmieszczenie procesow na   */
rep i = 0 for proc_no -1                            /* procesorach                  */
  place NetNode[i] on chip[i+1];
place from_server on host;
place to_server on host;
```

Wydruk 3. Program konfiguracyjny dla czterech procesorów połączonych w kwadrat



Rys. 4. Schemat połączeń procesorów

O stopniu złożoności tego języka niech świadczy zamieszczony na wydruku 3. opis konfiguracji programu składającego się z czterech procesów, wykonywanych na czterech procesorach połączonych w kwadrat (rys. 4). W węzłach 1, 2 i 3 umieszczono program z rysunku 3. Rysunek 5. przedstawia rozmieszczenie procesów i kanałów w węźle pierwszym.



Rys. 5. Rozmieszczenie procesów i kanałów w węźle 1 dla programu z rys. 3 i wydruku 3

Na płycie znajdują się cztery procesory T800, wyposażone w 2 MB pamięci każdy. Standardowe łącze *host* służy do komunikacji z komputerem centralnym. W sieci działają dwa rodzaje procesów. Umieszczony w węźle 0 proces *root* jest zdefiniowany inaczej niż pozostałe procesy *NetNode*, gdyż za pomocą kanałów *from_server* oraz *to_server* realizuje on operacje wejścia-wyjścia. Pozostałe, określone w opisie sieci, dwa kanały (*get* oraz *put*) służą do przesyłania danych między procesami zdefiniowanymi w różnych węzłach. Pozycja na liście parametrów procesu musi być zgodna z odwołaniami do kanałów w kodzie programu (funkcja *get–param* na rys. 3).

Kod wynikowy programów *root.c8s* i *node.c8s* zostaje przypisany do odpowiadających im procesów (instrukcja *use*). Procesy te zostają następnie rozmieszczone na poszczególnych procesorach (instrukcja *place*).

Programowanie systemów transputerowych jest zajęciem żmudnym. Przez cały czas należy mieć na uwadze konfigurację sprzętowo-programową. Niektóre łącza są zarezerwowane do specjalnych celów. Zmiana topologii sieci może wymagać zmian w programie, gdyż na jednym fizycznym łączu może być umieszczony jeden kanał. Ponieważ tylko pierwszy procesor w sieci może realizować operacje wejścia-wyjścia, wszystkie wyniki muszą być przesyłane przez węzły pośrednie, co utrudnia pisanie uniwersalnych programów lub wymaga stosowania dodatkowych mechanizmów, takich jak proponowany w dalszej części artykułu moduł komunikacyjny. Rozwiązaniem problemu przesyłania wiadomości między dowolnymi węzłami mogłaby być dynamiczna rekonfiguracja połączeń. Jednakże tylko jeden procesor nadzoruje pracę programowanego przełącznika, co prowadzi do takich samych trudności, jak przy komunikacji z jednostką centralną.

Moduł komunikacyjny dla sieci transputerowej

Proponowane przez autora rozwiązanie problemu przesyłania komunikatów między dowolnymi węzłami sieci komputerowej polega na wyposażeniu każdego węzła sieci w uniwersalny moduł komunikacyjny (ang. *router*), który realizuje operacje nadawania i odbioru dla umieszczonych w tym węźle procesorów oraz pośredniczy w wymianie komunikatów przeznaczonych dla innych wezłów. Z punktu widzenia programu użytkowego oferowane funkcje mają postać prostych wywołań bibliotecznych. Dodatkowo następuje uproszczenie budowy programu konfiguracyjnego, gdyż zostaje wyeliminowany opis połączeń między procesami.

Dobry algorytm przesyłania komunikatów powinien uwzględniać nie tylko poprawność transmisji, ale również chronić sieć przed wystąpieniem przeciążenia lub blokady. W celu sprostania tym dodatkowym kryterium, autor postanowił sięgnąć do rozwiązań stosowanych w sieciach komputerowych. Mówiąc dokładniej, mechanizmy sprzętowe transputera można zakwalifikować do warstw: fizycznej i łącza danych standardu ISO [10], na których proponowany router dobudowuje warstwy sieciową i transportową.

Przyjęto, że algorytm wyboru drogi nie będzie uwzględniał dynamicznych zmian połączeń między procesami i uszkodzeń węzłów, ponieważ rekonfiguracja płyty w trakcie działania programu jest bardzo trudna, a wystąpienie błędu powoduje zatrzymanie całej sieci.

W sumie realizacja routera sprowadza się do następujących zagadnień szczegółowych:

- nadawanie i odbiór wiadomości przez proces użytkowy (usługi warstwy transportowej);
- wybór drogi w sieci i zabezpieczenia przed przeciążeniami (usługi warstwy sieciowej);
- analiza topologii połączeń.

Udostępniane przez warstwę transportową operacje nadawania i odbioru wiadomości działają podobnie, jak standardowe w języku Parallel C funkcje wysyłania (*ChanOut*) oraz odczytu (*ChanIn*) informacji z kanału. Jedyna różnica polega na tym, że odczytywana jest wiadomość nadesłana nie przez konkretnego nadawcę, ale przez dowolny węzeł sieci.

Przyjęta przez warstwę sieciową wiadomość jest dzielona na pakiety, które są przesyłane między kolejnymi węzłami aż do procesu docelowego, gdzie zostają umieszczone w buforze wejściowym.

W ramach analizy topologii, na podstawie opisu sieci sprzętowej jest znajdowana najkrótsza droga między węzłami. Przygotowanie w każdym węźle tablice wyboru drogi służą do wyboru tej linii wyjściowej, którą zostanie wysłana informacja.

Warstwa sieciowa

W realizacji warstwy sieciowej są stosowane dwa modele przesyłania pakietów: model datagramowy oraz model połączeń wirtualnych.

W modelu datagramowym każda wiadomość jest transmitowana między kolejnymi punktami jako oddzielna jednostka. Wiadomości te mogą być wysyłane różnymi drogami oraz przybywać w kolejności różnej od kolejności wysłania (podobnie jak listy wysłane pocztą).

W modelu połączeń wirtualnych warstwa sieciowa przesyła pakiety między nadawcą i odbiorcą przez „doskonały" kanał (nie ma błędów, pakiety nadchodzą w kolejności ich nadania). Wewnętrzna struktura sieci nie zależy od modelu usług i dlatego jest możliwe stosowanie wewnątrz sieci datagramów, a na zewnątrz połączeń wirtualnych. Np. wewnątrz sieci ARPANET stosuje się model datagramowy, a użytkownik korzysta z usług połączenia wirtualnego. W sieci DECNET usługa datagramowa została zrealizowana przez zastosowanie datagramów wewnątrz sieci. Połączenia wirtualne wewnątrz sieci SNA służą do realizacji usług połączenia wirtualnego.

Do zalet połączeń datagramowych należy odporność na błędy i uszkodzenia węzłów i łączy. Możliwa jest również zmiana drogi w wypadku przeciążenia. Przy przesyłaniu niewielkiej ilości informacji połączenie datagramowe jest tańsze, gdyż brak jest tu fazy nawiązywania oraz zakończenia sesji.

Implementacja warstwy sieciowej jest wzorowana na sieci ARPANET. Warstwa dostarcza usług połączenia wirtualnego, realizowanego wewnętrznie jako połączenie datagramowe.

Zastosowanie połączeń datagramowych pozwala zaoszczędzić na kosztach zestawień połączeń, które są istotne przy małym rozmiarze komunikatów. Udostępnienie usług połączenia wirtualnego zwalnia projektantów wyższych warstw z obsługi błędów transmisji.

Wykorzystanie datagramów przy realizacji połączeń wirtualnych wymaga stosowania potwierdzeń między kolejnymi punktami sieci, gdyż w węźle pośrednim wszystkie bufory mogą być zajęte i nie ma miejsca na przyjęcie następnych pakietów. Wysłanie kolejnego pakietu nie nastąpi przed otrzymaniem potwierdzenia.

W sytuacji, gdy wysłana jest wiadomość wielopakietowa, należy zapewnić odpowiednią liczbę buforów w węźle docelowym. Gdyby warunek taki nie był spełniony, może nastąpić blokada. Wiadomość nie może być wysłana do procesu użytkowego, bo nie jest kompletna, a nie może być skompletowana, gdyż kolejne pakiety nie są przyjmowane z braku buforów. Dlatego też wysłanie wiadomości wielopakietowej poprzedza wysłanie pakietu żądania alokacji. Węzeł odbiorczy rezerwuje wówczas odpowiednią liczbę buforów lub – jeśli występuje brak wolnego obszaru – umieszcza żądanie w kolejce. Gotowość realizaji usługi jest sygnalizowana nadawcy przez wysłanie pakietu potwierdzenia alokacji. W przypadku wiadomości jednopakietowych, pakiet danych jest jednocześnie żądaniem przydziału, jeśli więc będzie brak buforu, to nastąpi odrzucenie pakietu, a żądanie rezerwacji buforu zostaje umieszczone w kolejce. W chwili gdy u odbiorcy zostanie zwolniony bufor, do nadawcy jest wysyłany pakiet potwierdzenia i nadawca retransmituje pakiet.

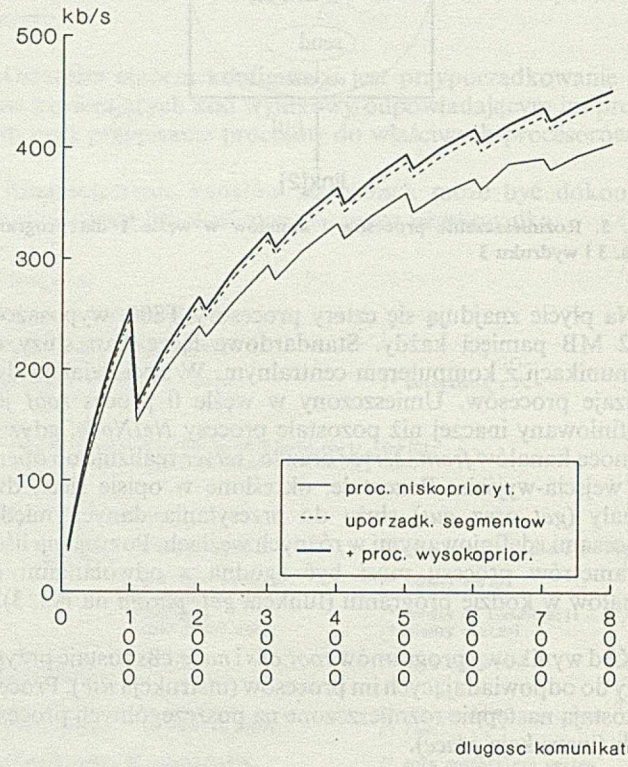
Dodatkowo ograniczono rozmiary kolejek wejściowych i wyjściowych. Rezerwacja bufora dla każdej linii wejściowej pozwala zabezpieczyć się przed sytuacją, gdy w węźle są zablokowane wszystkie bufory i nie jest nawet możliwe przyjęcie potwierdzenia nadejścia pakietu. Ograniczenie rozmiaru kolejki wyjściowej do określonego łącza umożliwia ruch na pozostałych liniach, gdy jeden z kanałów byłby przeciążony (zajęcie wszystkich buforów przez pakiety wysyłane w jednym kierunku zablokowałoby ruch w węźle). Reguła Irlanda [10] głosi, że maksymalny rozmiar kolejki m nie powinien przekraczać wartości $m = k/\sqrt{s}$, gdzie: k – liczba buforów, s – liczba linii wyjściowych.

Wyniki działania programu

Router został przetestowany dla sieci o różnych topologiach, rozmiarach pakietów i różnej liczbie buforów wejściowych. Obserwowano również wpływ uporządkowania segmentów i zastosowania procesów wysokopriorytetowych na szybkość transmisji.

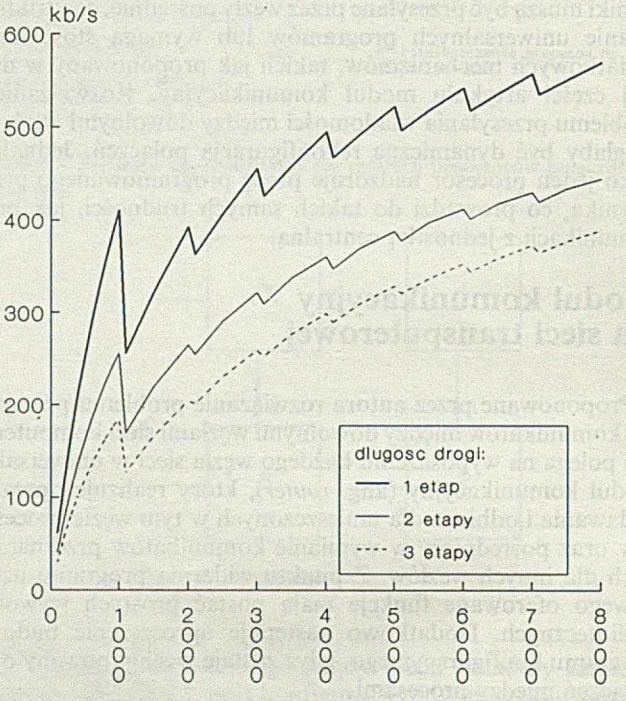
Ze względu na konieczność ograniczenia rozmiaru buforów, maksymalna długość wiadomości została ograniczona do ośmiu pakietów. Autor badał szybkość transmisji dla rozgłaszania komunikatów (ang. *broadcasting*) oraz komunikacji typu „każdy z każdym".

Na wstępie autor analizował wpływ zmiany kolejności segmentów, dzięki której alokowane dynamicznie bufory wejściowe i wyjściowe, bufory oraz kolejki danych mogą zostać umieszczone w szybkiej pamięci statycznej (4 kb wewnątrz procesora oraz 28 kb w pamięci zewnętrznej). Uzyskano przyspieszenie transmisji o około 10%. Zaimplementowanie reoutera jako zbioru procesów wysokopriorytetowych spowodowało dalszy wzrost szybkości przesyłania danych rzędu 3–4%. Wynika to z krótszego czasu oczekiwania procesów wysokopriorytetowych na dostęp do kanału.



Rys. 6. Wpływ uporządkowania segmentów i zastosowania procesów wysokopriorytetowych na średnią szybkość rozgłaszania komunikatów

Rys. 6 przedstawia średnią szybkość rozgłaszania w sieci transputerów połączonych w łańcuch dla pakietów 1000-bajtowych.



Rys. 7. Szybkość rozgłaszania komunikatów dla pakietów o rozmiarze 1000 bajtów

Następne testy wykonywano wyłącznie dla routera zaimplementowanego jako zbiór procesów wysokopriorytetowych i obszaru sterty, umieszczonego w pamięci statycznej. Na rysunku 7 przedstawiono szybkość transmisji do węzła sąsiedniego i węzłów odległych o jeden lub dwa węzły pośrednie.

Zaobserwowano, że zwiększenie rozmiaru pakietu istotnie wpływa na zwiększenie szybkości przesyłania. Narzut związany z obsługą danych wewnątrz węzła jest większy niż koszt przesłania danych, które po inicjalizacji jest realizowane współbieżnie z pracą procesora. Zwraca uwagę znaczny spadek szybkości transmisji przy wysyłaniu niepełnych bloków, co należy uwzględnić przy wyborze długości pakietu.

Wprowadzony mechanizm istotnie rozszerza możliwości oferowane przez sprzęt, pozwalając na komunikację między dowolnymi węzłami w sieci. Nawet dla stosunkowo krótkich pakietów, szybkość transmisji wynosi ok. 400 kb i zmniejsza się jedynie o 25% po przejściu przez każdy węzeł pośredni. Biorąc pod uwagę maksymalną teoretyczną przepustowość sieci 1,75 Mb [4] – mierzoną w idealnych warunkach, gdy procesory nie wykonują innych czynności i cały program mieści się w pamięci statycznej procesora – uzyskane wyniki należy uznać za dobre.

meditronik SPÓŁKA z o.o.

UMC UNITED MICROELECTRONICS CORPORATION

HEWLETT PACKARD COMPONENTS

BOURNS

- CZĘŚCI ELEKTRONICZNE
- KOMPUTERY PS/1,PS/2
- DRUKARKI HP
- INSTALACJE SIECI KOMPUTEROWYCH

- UKŁADY PAMIĘCI
- UKŁADY KOMPUTEROWE
- UKŁADY KOMUNIKACYJNE I KOMERCYJNE

- TRANSOPTORY
- WSKAŹNIKI ŚWIETLNE
- WYŚWIETLACZE LED
- PRODUKTY KODÓW KRESKOWYCH
- KONTROLERY I CZUJNIKI RUCHU
- TECHNIKA ŚWIATŁOWODOWA
- ELEMENTY W.CZ. I MIKROFALOWE
- PODZESPOŁY DO MONTAŻU POWIERZCHNIOWEGO (SMD)

- POTENCJOMETRY TRIMPOT
- HYBRYDY REZYSTOROWE
- REZYSTORY SUBMINIATUROWE
- BEZPIECZNIKI MULTIFUSE
- POTENCJOMETRY PRECYZYJNE
- POTENCJOMETRY PANELI CZOŁOWYCH I KODERY
- CEWKI I TRANSFORMATORY
- CZUJNIKI CIŚNIENIA, POŁOŻENIA I PRZYŚPIESZENIA

Partnerzy handlowi: ANALOG DEVICES, ITT, MOTOROLA, SAMSUNG, TELEFUNKEN i inni

PRZEDSTAWICIELSTWO

DYSTRYBUCJA

PMI

DYSTRYBUCJA

IBM Business Partner

meditronik sp. z o.o.

00-194 Warszawa,ul. Dzika 4
tel. (02) 6352263, 6352264
fax (02) 6352195, tlx 816075

SO/575/91

Uwaga!
INFORMATYKĘ już można zaprenumerować także na poczcie.

Szczegóły na II stronie okładki i stronie 28.

W artykule wykorzystano niektóre wyniki badań przeprowadzonych w ramach pracy magisterskiej „Uniwersalny moduł komunikacyjny dla sieci transputerowych", napisanej w Instytucie Informatyki Uniwersytetu Warszawskiego oraz doświadczenia zdobyte w czasie pobytu w Sheffield City Polytechnic.

W tym miejscu chciałbym podziękować dr. Jarosławowi Milewskiemu za inspirację w wyborze tematu pracy, cenne wskazówki przy przygotowywaniu koncepcji badań i pomoc w opracowaniu ich wyników oraz prof. Antoniemu Kreczmarowi za okazaną mi życzliwość podczas realizacji projektu.

LITERATURA

[1] Błaszczak S.: Język Occam. Informatyka 6, 7, 1988
[2] Caban D.: Transputer – narzędzie programowania równoległego. Informatyka 6, 1988
[3] Hoare C.A.R.: Communicating sequential processes. Comm. of the ACM 21(8), 1978, pp. 666–677
[4] INMOS Ltd.: Transputer Databook. INMOS 1989
[5] INMOS Ltd.: ANSI C toolset user manual. INMOS 1990
[6] INMOS Ltd.: ANSI C toolset reference manual. INMOS 1990
[7] INMOS Ltd.: B008 support software. INMOS 1990
[8] INMOS Ltd.: IMS B008 User guide and reference manual. INMOS 1990
[9] INMOS Ltd.: The T900 Transputer Products Overview Manual. INMOS 1991
[10] Tanenbaum A.: Sieci komputerowe, WNT 1988.

Nowe książki

WYDAWNICTWA NAUKOWO-TECHNICZNE

Seria: Podręczna Pamięć Programisty

Krzysztof Kamiński: Turbo Pascal 5.0. WNT, Warszawa 1992 r., wyd. 3, s. 96, format B6. ISBN 83-204-1508-X

Opracowanie jest przeznaczone dla osób znających język Pascal. Zawiera krótkie informacje dotyczące dostarczanych przez producenta funkcji i procedur oraz zasady posługiwania się programem zawartym w pliku TURBO. EXE, integrującym edytor, kompilator i program uruchomieniowy Turbo Pascala 5.0. Opisy procedur i funkcji podają rodzaje typów możliwych do wykorzystania w konfiguracji sprzętowej bez koprocesora arytmetycznego.

Wiesława Zapart: Norton Commander wersja 3.0. WNT, Warszawa 1992 r., wyd. 1, s. 96, format B6. ISBN 83-204-1535-7

Program Norton Commander wersja 3.0 opisany w książce jest nakładką na system operacyjny DOS, umożliwiającą realizowanie poleceń systemu operacyjnego w uproszczony dla użytkowników sposób. Nie wymaga znajomości składni poleceń DOS-a. Dzięki swojej wielofunkcjonalności jest bardzo przydatny w pracy z katalogami dyskowymi.

Beata Radojewska, Sergiusz Paleta: AutoCAD (obejmuje wersje do AutoCAD-a 10 włącznie). WNT, Warszawa 1992, wyd. 1, s. 140, format B6. ISBN 83-204-1558-6

AutoCAD (nazwa zastrzeżona) jest programem służącym do tworzenia rysunków i projektowania wspomaganego komputerem. Znajduje zastosowanie w technice, architekturze i sztuce. Przedstawiony opis programu dotyczy wersji AutoCAD-a pracujących z komputerami zgodnymi z IBM, z systemem operacyjnym DOS. Minimalna konfiguracja potrzebna do działania programu to: 512 KB pamięci RAM (640 KB, jeśli używa się opcji AutoLISP), jedna stacja dysków i monitor z kartą graficzną. W książce przedstawiono dokładne odwzorowania napisów i komunikatów, które mogą pojawić się na ekranie w trakcie pracy. AutoCAD jest wyposażony w kompilator języka LISP.

LUCYNA KORBEL
Katedra Informatyki
ELŻBIETA MIŁOSZ
Katedra Zarządzania
MAREK MIŁOSZ
Katedra Informatyki
Politechnika Lubelska

Systemy zarządzania relacyjnymi bazami danych oraz ich przydatność do budowy informatycznych systemów zarządzania

Po złagodzeniu w 1990 r. ograniczeń COCOM do Polski zaczęły docierać produkty zaawansowanej technologii informatycznej – w tym również oprogramowania. Ograniczenia te dość skutecznie „chroniły" polskich informatyków przed możliwością praktycznego zapoznania się z całą gamą nowoczesnego oprogramowania. Do Polski nie dochodziły nawet informacje o poszczególnych systemach. Embargo obejmowało także dostęp do parametrów handlowych i reklamę. Do szczególnie chronionych przez COCOM należały nowoczesne, wysokowydajne systemy zarządzania relacyjnymi bazami danych.

W tych warunkach rewelacyjnymi systemami zarządzania relacyjną bazą danych (SZRBD) okazały się dBase oraz Clipper. Systemy te, przeznaczone w zasadzie do informatyzacji małych firm i przedsiębiorstw, zaczęto używać do budowy dużych, kompleksowych i zintegrowanych systemów informatycznych zarządzania (SIZ). Znane są nawet próby tworzenia za pomocą Clippera lokalnych sieci mikrokomputerów typu IBM PC (np. Novell Netware) systemów bankowych.

Pomimo znacznych nakładów sił i środków (a także niemalże genialności polskich programistów), w większości przypadków nie udały się próby tworzenia dużych SIZ – systemy takie nie osiągały założonej efektywności i wydajności działania. Częste przestoje i awarie systemu, zwiększające się nakłady na konserwacje, rozwój i integrację różnych podsystemów, a także niemożliwe do przekroczenia bariery ilościowe zniechęcają największych nawet sympatyków informatyzacji.

Istnieje obiektywna konieczność zmiany narzędzi informatycznych używanych do tworzenia SIZ na bardziej nowoczesne, wolne od wskazanych wad, wysokowydajne i otwarte na rozwój platform sprzętowych.

Systemy zarządzania relacyjnymi bazami danych, jakie ostatnio pojawiły się na rynku polskim są mało znane, brak jest jakichkolwiek szerszych doświadczeń w ich eksploatacji. Dlatego też na obecnym etapie o doborze takiego czy innego systemu decyduje często przypadek.

Niniejszy artykuł zawiera informacje dotyczące kilku wybranych systemów relacyjnych baz danych (ORACLE, PROGRESS, INFORMIX, INGRES) oraz porównanie tych systemów pod kątem ich przydatności do budowy systemów informatycznych zarządzania.

Podstawą opracowania są informacje z różnych publikacji, będących wyciągami z dokumentacji systemów, publikacje zawierające doświadczenia i uwagi użytkowników oraz informacje zdobyte w trakcie Szkół Polskiego Towarzystwa Informatycznego w Świnoujściu i Szczyrku, a także II Forum Technologii Informatycznych w Mrągowie.

W artykule wykorzystano rezultaty pracy naukowo-badawczej: „Wykorzystanie nowoczesnych narzędzi projektowo-programowych w konstruowaniu systemów informatycznych dla transportu samochodowego" (nr KBN 9 9498 91 02).

Kryteria użyteczności relacyjnych baz danych w aspekcie budowy SIZ

Ocenę użyteczności systemu zarządzania relacyjnymi bazami danych przeprowadza się zwykle w wielu płaszczyznach, poddając ocenie:

- parametry techniczne,
- jakość dokumentacji użytkowej,
- zakres wspomagania programistów przez firmę (dealera),
- stopień rozpowszechnienia w świecie.

Oczywiście ogólna ocena systemu powinna być dokonana z uwzględnieniem obszaru przyszłych zastosowań bazy danych (w szczególności typu i wielkości SIZ) oraz relacji cenowych.

Parametry techniczne SZRBD należą w zasadzie do parametrów ilościowych i dlatego stosunkowo łatwo poddają się ocenie. Do podstawowych parametrów technicznych można zaliczyć:

- środowisko pracy bazy danych (różnorodność platform sprzętowych, systemów operacyjnych, pracy w sieciach, możliwość tworzenia rozproszonych baz danych, współpraca z systemami okien i możliwość odczytywania informacji z innych baz danych);

- architekturę systemu bazy danych (przetwarzanie w architekturze klient-serwer, wielowątkowe i rozproszone);
- organizację bazy danych (ograniczenia ilościowe: długość pola, rekordu, tablicy i bazy; modyfikowalność bazy danych: restrukturyzacja i reorganizacja; sposób i efektywność indeksowania; środki zabezpieczenia integralności bazy danych; obecność narzędzi do interaktywnej pracy z bazą danych);
- sprawność: czasową (szybkość realizacji wybranych klas zadań użytkowych i prac technicznych typu indeksowanie czy archiwizacja) oraz pamięciową (zajętość pamięci operacyjnej przez system uruchomieniowy i wybrane, porównywalne aplikacje; zajętość pamięci dyskowej przez bazy danych o identycznej zawartości informacyjnej);
- mechanizmy współbieżnej pracy (ochrona przed spadkiem wydajności systemu przy wzroście liczby użytkowników; sposób rozwiązywania konfliktów współdzielenia zasobów: rezerwacja–blokowanie bazy danych, pliku, strony czy rekordu, obecność mechanizmów wykrywania i rozstrzygania konfliktów),
- ochronę bazy danych (przed niepowołanym dostępem: identyfikacja praw użytkownika do pliku, rekordu, pola i(lub) procedury; przed utratą spójności bazy: obecność dwu – lub więcejfazowego protokołu potwierdzeń; łatwość i efektywność systemu składowania i odtworzenia informacji z bazy danych),
- język obsługi bazy danych (zawartość i przejrzystość zbioru poleceń języka czwartej generacji – 4GL; zgodność ze standardem języka SQL; możliwość łączenia z innymi językami: C, COBOL, Pascal; przenośność; porównywalność poleceń języka do poleceń już istniejących);
- obecność i możliwości polskiej wersji językowej (polskojęzyczny interfejs, indeksowanie plików i komunikaty systemowe w języku polskim).

Jakość dokumentacji użytkowej obejmuje następujące parametry:
- dostępność polskojęzycznej dokumentacji systemu, języka i opisu zastosowań;
- przejrzystość dokumentacji dostarczanej z systemem, układ treści i sposób jej przedstawienia;
- łatwość instalowania systemu;
- dostępność do literatury uzupełniającej.

Wspomaganie programistów i użytkowników przez firmę może być ocenione za pomocą następujących elementów:
- całodobowy serwis i „gorąca" linia dla programistów (w Polsce i po polsku);
- organizacja, dostępność i częstotliwość szkoleń (z uwzględnieniem różnych poziomów wiedzy użytkownika i programistów);
- usuwanie usterek przez firmę (bezpłatność wersji rozwojowych, sposób usuwania błędów);
- obecność w systemie programów wspomagających (narzędzi do zakładania bazy danych i jej kontroli, budowy prototypów systemu, tworzenia układów menu, pomocy kontekstowej, bibliotek gotowych procedur, generatorów kodu źródłowego programu, programów śledzących i monitorujących wykonanie programu);
- dostępność systemów CASE wspomagających wytwarzanie aplikacji w języku danej bazy;
- otwartość firmy lub dealerów w stosunku do użytkowników (organizacja klubów użytkowników, wydawanie gazetek, biuletynów, udział firmy w różnorodnych imprezach środowiska informatyków – konferencjach, pokazach, targach);
- ocena baz danych przez ich użytkowników (poziom zadowolenia programistów, administratorów systemów informatycznych i końcowych użytkowników aplikacji).

Kryterium rozpowszechnienia w świecie to przede wszystkim liczba i dynamika sprzedaży pełnych wersji systemu oraz liczba dostępnych (używanych) aplikacji. Do tego kryterium należy zaliczyć także plany rozwojowe firmy (z oceną możliwości ich realizacji). Stopień rozpowszechnienia do niedawna był bardzo ważną cechą bazy danych. Szczególnie duże znaczenie miał w przypadku budowy systemów informatycznych mających współpracować w sposób bezpośredni z innymi systemami o zasięgu ogólnokrajowym (np. z systemem ewidencji ludności) czy też światowym (sieci bankowe). W chwili obecnej znaczenie tego kryterium zmalało. Jest to związane z rozwojem systemów otwartych z jednej strony, i możliwością prostej konwersji formatów danych – z drugiej. Ponadto większość współczesnych systemów zarządzania relacyjnymi bazami danych ma środki do komunikowania się z innymi bazami.

Podczas rozpatrywania możliwości zastosowań danego SZRBD do budowy konkretnego SIZ uwzględnia się przede wszystkim następujące jego parametry:
- środowisko funkcjonowania SIZ,
- wielkość bazy danych i stopień złożoności realizowanych zadań,
- rozproszoność bazy danych,
- liczbę jednoczesnych użytkowników systemu,
- konieczność zapewnienia ochrony danych,
- wartość aplikacji.

Przegląd własności wybranych SZRBD

ORACLE

System zarządzania relacyjną bazą danych ORACLE może pracować na dużych komputerach, stacjach roboczych, minikomputerach i mikrokomputerach. Wykorzystuje on język ORACLESQL, który jest nadzbiorem języka SQL.

Jest to bardzo efektywny, wielodostępny system o takich zaletach, jak: przenośność, blokowanie na poziomie rzędu (rekordu), optymalizacja operacji wejścia–wyjścia, odporność na awarie, odpowiednie zabezpieczenie przed nieupoważnionym dostępem oraz dobry marketing. Dostęp do bazy danych jest możliwy w trybie konwersacyjnym lub wsadowym.

ORACLE jest systemem dopuszczającym współbieżną pracę wielu użytkowników. Są stosowane następujące mechanizmy blokad: wewnętrzne, słownika danych, tabel (plików) i wierszy (rekordów).

Baza danych ORACLE składa się z trzech rodzajów plików:
- plików bazy danych (od 1 do 255),
- plików dziennika (minimum 2),
- plików informacyjnych (minimum 1).

W zakresie funkcjonalności SZRBD ORACLE oferuje stosunkowo mało. Np. w wielu systemach sprawdzanie integralności danych można zdefiniować tylko raz w bazie danych, natomiast w tym systemie programiści muszą pamiętać, aby robić to w swoich programach.

Efektywność ORACLE zależy od wielkości i złożoności bazy danych. Dla większości systemów umiarkowanej wielkości (do 1 GB) z 20 do 30 użytkownikami ORACLE doskonale nadaje się do zastosowania. Nie są znane możliwości zastosowań ORACLE w środowiskach wieloplikowych ze 100 lub więcej użytkownikami pracującymi w trybie on-line. Ekstensywne buforowanie także zmniejsza osiągi systemu ORACLE.

ORACLE ma bogaty zestaw ułatwiających dostęp do bazy danych pakietów narzędziowych, m.in. pakietów:
- bezpośredniego dostępu do bazy danych (SQL*Plus),
- interakcyjnego wspomagania użytkownika w generowa-

niu formatek aplikacji, bez potrzeby pisania programów (SQL*Forms),
- projektowania oraz generowania dynamicznych menu, upraszczających obsługę tworzonej aplikacji (SQL*Menu),
- obsługi interakcyjnie żądanych raportów, tworzonych na podstawie informacji zawartych w bazie danych (SQL*Report),
- umożliwiających zaprojektowania złożonych raportów, bez konieczności pisania programu (SQL*ReportWriter),
- kompilatora języka dBase III Plus (ORACLE for dBase Interpreter Compiler),
- konwersji plików z danymi w innych formatach (np. Lotus 1-2-3, dBase III Plus) do formatu bazy danych ORACLE (SQL*Loader),
- arkusza kalkulacyjnego (SQL*Calc),
- graficznej prezentacji informacji generowanych w postaci raportów (SQL*Graph),
- umożliwiających komunikację między rozproszonymi bazami danych z wykorzystaniem różnych protokołów komunikacyjnych, takich jak: DECnet, TCP/IP, SNA, Novell SPX, Named Pipes, itd. (SQL*Net),
- umożliwiających współpracę z innymi bazami danych, np. z DB2, SQL/DS i RMS (SQL*Connect),
- tworzących zintegrowany otwarty system rozproszonej bazy danych, rozmieszczonej na komputerach o różnej architekturze i różnych systemach operacyjnych (SQL*Star).

System ORACLE dopuszcza pisanie programów z zastosowaniem powszechnie znanych języków programowania, takich jak np. C, PASCAL, FORTRAN, COBOL, ADA. Istnieją dla tych języków pakiety prekompilatorów (PRO*C, PRO*PASCAL, itd.). Program może być pisany w jednym z tych języków z pełnym wykorzystaniem instrukcji języka SQL, włączonym bezpośrednio w tekst programu. Po skompilowaniu program taki może funkcjonować tylko z samym SZRBD ORACLE.

Istnieją także gotowe pakiety wspomagające projektowanie i implementację bardziej złożonych aplikacji, stosujące standardową metodę inżynierii programowania. Należą do nich CASE*Designer, CASE*Dictionary oraz CASE*Generator.

Istniejące oprogramowanie usługowe systemu ORACLE pozwala administratorowi bazy danych wykonać szereg funkcji, włącznie z monitorowaniem osiągów i użytkowania, otwieraniem, zamykaniem, składowaniem i naprawą bazy danych.

Obsługa systemu ORACLE wymaga średnich umiejętności, ale wysokie kwalifikacje umożliwiają opracowanie bardziej efektywnych aplikacji.

W zakresie osiągów użytkownicy oceniają system ORACLE jako przeciętny, a nawet słaby, w przypadku systemów przetwarzania transakcji pracujących w trybie on-line na sprzęcie IBM. System ORACLE charakteryzuje się doskonałą przenośnością i dobrą współbieżnością, natomiast funkcjonalność bazy danych jest oceniana jako słaba. Funkcjonalność rozproszonego systemu ORACLE ocenia się jako przeciętną.

PROGRESS 4

PROGRESS 4GL i SZRBD jest narzędziem służącym do zarządzania dużą liczbą danych w rozproszonym i niejednorodnym środowisku systemowym. System może być instalowany na różnych (ponad 200) platformach sprzętowych np. Apple, Hewlett Packard, IBM, ICL, Philips, Siemens i innych.

PROGRESS pracuje pod kontrolą takich systemów operacyjnych jak DOS, OS/2, UNIX i jego pochodne (XENIX, ULTRIX, AIX, ...), VMS, BTOS, CTOS. Może pracować praktycznie w każdym środowisku sieciowym. Ma architekturę klient–serwer z możliwością pracy wieloserwerowej i wielowątkowej.

Do tworzenia aplikacji służą takie języki, jak PROGRESS 4GL i PROGRESS SQL, chociaż można używać także języków C, COBOL i PASCAL.

PROGRESS ma cechy pozwalające na współpracę z systemami okien, takimi jak: MOTIF, MS Windows 3.0, Open Look, Open Desktop, X Windows, DEC Windows, Presentation Manager oraz NewWawe (HP).

System PROGRESS jest wyposażony w mechanizm, który umożliwia wygodną wymianę danych z otoczeniem. Możliwe jest pobieranie oraz wysyłanie danych z poziomu aplikacji z/lub do plików ASCII o określonym formacie. W czasie pracy interaktywnej można wysyłać dane w formacie ASCII, DIF, SYLK, co umożliwia wymianę danych z takimi pakietami, jak Lotus 1-2-3, Excel, Uniplex, Multiplan, 20/20, WordPerfect, WordStar, Word oraz BTOS Office Writer.

W systemie istnieją opcje pozwalające na import danych w formacie DIF, SYLK, ASCII, a także na import definicji struktur oraz danych bezpośrednio z bazy DBASE (II, III, III+, IV). Istnieje również możliwość wykonania niektórych operacji na bazach innych SZRBD (AS/400, ORACLE, RDB, RMS, C-ISAM).

PROGRESS ma cechę pozwalającą stworzyć całość aplikacji w 4GL, bez konieczności odwoływania się do języków trzeciej generacji. Taka możliwość istnieje i jest wykorzystywana, nawet wówczas gdy system pracuje w nietypowym otoczeniu. W systemie możliwe jest blokowanie na poziomie wierszy (rekordów) lub całych tablic (plików). Blokowanie odbywa się w sposób automatyczny.

Integralność danych jest uzyskiwana przez:
- ochronę przed niepowołanym dostępem (hasła i uprawnienia dostępu do plików i pól, a także do opcji modyfikacji struktury bazy danych),
- blokadę określonych operacji (z poziomu aplikacji),
- zapewnienie spójności logicznej podczas wprowadzania lub usuwania informacji (na poziomie słownika bazy danych),
- ochronę przed uszkodzeniem lub utratą danych (dwufazowy protokół potwierdzeń, przechowywanie obrazu pierwotnego),
- zabezpieczenie obrazu końcowego (więcej niż jedna wersja pliku wynikowego),
- możliwość składowania stanu całej bazy lub tylko części modyfikowanej, bez przerywania pracy systemu lub z przerwaniem.

PROGRESS nie ma własnego środowiska CASE. Takie narzędzia, jak np. FAST TRACK, pomagają w tworzeniu aplikacji, lecz nie jest to środowisko CASE w pełnym tego słowa znaczeniu.

INFORMIX

Większość pakietów INFORMIXa może pracować pod takimi systemami operacyjnymi, jak UNIX, XENIX, MS-DOS, OS/2, VMS, jak również w środowisku sieci lokalnych ze standardowymi protokołami, np. z TCP/IP, Novell NetWare, itp. Językiem stosowanym w programach, tak użytkowych jak i systemowych, jest język zapytań klasy SQL będący rozszerzeniem standardu.

W niektórych pakietach (INFORMIX-SQL) jest stosowany też język Query By Example. Oprócz tego system INFORMIX oferuje: wysokopoziomowy język programowania 4GL, języki

definiowania formularzy i tworzenia raportów oraz języki będące rozszerzeniami języków trzeciej generacji (C, COBOL, FORTRAN, ADA) o instrukcje języka SQL.

W systemie INFORMIX istnieje sposób optymalizacji procesu dostępu do danych. W słowniku bazy danych można wyróżnić dwa poziomy: systemowe tablice opisu bazy danych oraz tablice związane z pracą z formularzami i budowaniem menu użytkownika.

INFORMIX w wersji 4.10 ma pakiet typu CASE, ułatwiający tworzenie oprogramowania aplikacyjnego. Jest to system wieloprocesorowy i wielodostępny. Z wielodostępem jest związany mechanizm blokowania dostępu do danych na poziomie wierszy, tablic lub na poziomie całej bazy danych.

System INFORMIX realizuje procesy aktualizacji bazy danych bez tworzenia kopii aktualizowanego fragmentu, co zwiększa czas trwania blokad i obniża jego efektywność.

W systemie tym istnieją dwa mechanizmy służące do odtwarzania stanu bazy danych. Podstawową konstrukcją jest plik transakcyjny, wykorzystywany w czasie odtwarzania stanu całej bazy, a oprócz tego istnieje możliwość kronikowania zmian w poszczególnych tablicach. Tworzenie kopii bazy danych i kolejnych plików transakcyjnych jest możliwe po wyłączeniu współbieżnego przetwarzania bazy danych, a więc paktycznie po unieruchomieniu wszystkich aplikacji (systemu użytkowego).

INGRES

INGRES działa na ponad 40 różnych platformach sprzętowych pod kontrolą różnych systemów operacyjnych, takich jak np. VME, UNIX, OS/2, MS-DOS, AIX, SUNOS, HP-UX, DEC ULTRIX, SCO UNIX..

Do najważniejszych zalet tego systemu należy zaliczyć:
- wybitną funkcjonalność,
- bogactwo struktur danych,
- wieloserwerową architekturę,
- wyrafinowany optymalizator.

Wybitną funkcjonalność INGRES zawdzięcza przede wszystkim dwóm systemom: zarządzania wiedzą i zarządzania obiektami. System zarządzania wiedzą umożliwia utworzenie bazy wiedzy zawierającej reguły, sterowanie zasobami systemu oraz sterowanie dostępem do baz danych. Liczba reguł i procedur, które można związać z tablicami jest nieograniczona. System zarządzania obiektami umożliwia zdefiniowanie własnych typów danych oraz funkcji i operatorów charakteryzujących zdefiniowane obiekty. Definiowanie odbywa się tylko raz i mogą z niego korzystać wszyscy użytkownicy, co ułatwia proces tworzenia oprogramowania aplikacyjnego oraz poprawia efektywność systemu. Zarządzanie wiedzą i obiektami daje INGRES-owi istotną przewagę nad innymi SZRBD.

Architektura INGRES-a jest wielowątkowa oraz wieloserwerowa i wykorzystuje wszystkie możliwości przetwarzania współbieżnego dostępnego na systemach wieloprocesorowych i systemach sieciowych.

INGRES ma możliwości przetwarzania rozproszonego, obejmujące zarówno rozproszenie przetwarzania, jak i rozproszenie danych. Użytkownicy nie tylko mogą wymieniać dane między rozproszonymi bazami danych INGRES-a, ale również korzystać z danych zawartych w bazach danych innych systemów, takich jak np.: DB2, IMS, RDB, RMS, dBASE III, HP Allbase, Data General. Dostęp i manipulacja danymi są obsługiwane za pomocą języka OPEN SQL zawierającego około 85% możliwości języka SQL..

Optymalizator zapytań INGRES-a jest oparty na bazie analizy kosztów i wykorzystuje statystykę zawartą w katalogach słownika danych. Zabezpieczenie integralności danych umożliwiają m.in.: reguły integralności zawarte w bazie, metody sładowania danych, system blokad oraz system rejestrowania użytkowników.

Blokowanie może być wykonywane na dwóch poziomach: tablicy lub strony i może przebiegać wg jednego z następujących dwóch sposobów: blokowania dzielonego – dla odczytu lub blokowania wybiórczego (wyłącznie) – dla zapisu. Niestety niedostępny jest mechanizm blokowania na poziomie wiersza (rekordu).

INGRES ma wielopoziomową kontrolę uprawnień dostępu na poziomach: administratora systemu INGRES, administratora bazy danych oraz użytkownika.

Środowisko programowe INGRES-a pozwala całkiem dobrze pracować użytkownikom bez głębokiej wiedzy informatycznej, jednak dla większych systemów niezbędni są specjaliści.

Środowisko bazy danych INGRES-a obejmuje także narzędzia służące do analizy i projektowania systemów aplikacyjnych o nazwie INGRES/teamwork I-CASE.

INGRES umożliwia wykorzystanie niektórych swoich produktów na komputerach osobistych (VisionPro oraz Embedded SQL dla języka C).

Efektywność INGRES-a dla małych i dużych systemów zapytań oraz dla małych systemów działających w trybie on-line jest dobra, natomiast dla dużych systemów OLTP jest ona oceniana przez użytkowników jako średnia. Funkcjonalność bazy danych użytkownicy określają jako bardzo dobrą, a jego przenośność – jako dobrą.

Metodologia kompleksowej porównawczej oceny baz danych

Zakres porównania obejmuje te cechy relacyjnych systemów baz danych, które wydają się być istotne w zastosowaniach do budowy SIZ. Dostępne informacje dotyczące porównywanych systemów baz danych ograniczyły dobór cech (nie można uwzględniać cech, których wartości nie są znane).

Metodologia kompleksowej oceny porównawczej SZRBD składała się z następujących etapów:
- wybór zbioru parametrów SZRBD – cech, określających jakość bazy danych w aspekcie zastosowań do budowy SIZ,
- określenie wartości cech dla systemu wzorcowego (idealnego), a także wag poszczególnych parametrów w ocenie kompleksowej (metoda ekspertowa),
- porównanie kompleksu cech baz danych z parametrami systemu wzorcowego metodą taksonomii wrocławskiej,
- graficzne przedstawienie rezultatu porównania w postaci dendrytu z wyjawieniem skupień.

System wzorcowy jest określany przez maksymalne wartości cech – parametrów jakości. Do zobrazowania cech systemów zastosowano postać graficzną, która jest dendrytem pokazującym odległości poszczególnych systemów między sobą, a także ich odległość od systemu wzorcowego.

W opracowaniu dendrytu została wykorzystana metoda taksonomiczna znana jako taksonomia wrocławska [4]. Poszczególnym cechom, zależnie od ich wpływu na proces projektowania, zostały nadane wagi.

Większość cech przyjmuje wartości jakościowe. W stosowanej metodzie wartości jakościowe cech zastąpiono wartościami liczbowymi określonego typu, które następnie po znormalizowaniu [5] wykorzystano do obliczenia odległości między poszczególnymi systemami według wzoru:

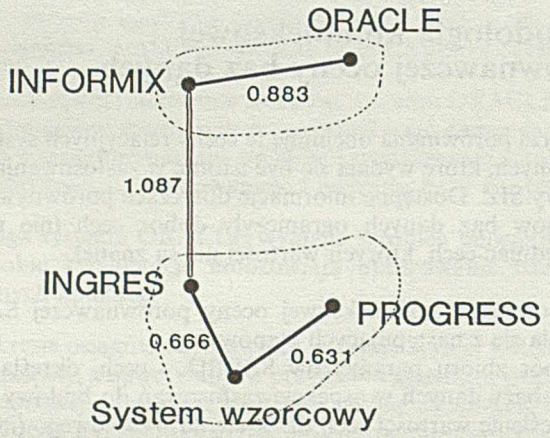
$$d_{ij} = \sqrt{\sum_{k=1}^{N} w_k (y_{ik} - y_{jk})^2}, \quad \text{dla } i, j = 1, ..., Q$$

gdzie:
Q – liczba porównywalnych systemów z wzorcowym włącznie;
N – liczba cech;
w_k – waga k-tej cechy;
y_{ik} – znormalizowana wartość k-tej cechy dla i-tego systemu: $y_{ik} = x_{ik}/M_k$
M_k – maksymalna wartość k-tej cechy;
X_{ik} – wartość k-tej cechy dla i-tego systemu.

Metoda konstruowania dendrytu składa się z dwóch kroków:

● Dla każdego porównywanego systemu jest wyszukiwany system leżący najbliżej danego i łączy się je odcinkiem. Po połączeniu wszystkich systemów z najbliższymi otrzyma się skupienia pierwszego rzędu, które mogą już tworzyć dendryt kompletny, nazywany dalej dendrytem. Jeśli jednak istnieje więcej niż jedno skupienie pierwszego rzędu, wówczas należy utworzyć skupienia wyższego rzędu.

● Skupienie wyższego rzędu otrzymuje się łącząc każde skupienie danego rzędu ze skupieniem rzędu najbliższego względem niego. Odcinkiem łączącym skupienie danego rzędu z rzędem najbliższym jest odcinek łączący system należący do danego skupienia z najbliższym systemem nie należącym do tego skupienia. Jeśli po połączeniu wszystkich skupień danego rzędu ze skupieniami najbliższymi otrzyma się więcej niż jedno skupienie wyższego rzędu, procedurę łączenia powtarza się dotąd, aż otrzyma się jedno skupienie stanowiące dendryt.



Dendryt kompletny dla porównywanych systemów zarządzania relacyjnymi bazami danych

Rezultaty analizy porównawczej

Na podstawie analizy sposobu projektowania, oprogramowywania, wdrażania i konserwacji SIZ dokonano doboru parametrów – cech (i ich wag) do porównania poszczególnych SZRBD (tabela 1). Wartości (nieznormalizowane) poszczególnych cech dla wszystkich pięciu systemów zostały podane w tabeli 2.

Tabela 3 zawiera macierz wzajemnych odległości poszczególnych SZRBD i systemu wzorcowego, natomiast rysunek przedstawia przybliżony wygląd dendrytu, zbudowanego na podstawie wyznaczonych odległości.

Tabela 1. Oznaczenia, wagi i zakres wartości poszczególnych cech SZRBD

Cecha	Symbol	Waga	Zakres wartości
Różnorodność platform	x_1	0,8	1–10 systemów
Przenośność oprogramowania	x_2	0,9	1–5 pkt.
Praca współbieżna	x_3	1,0	1–5 pkt.
Ochrona danych	x_4	1,0	1–5 pkt.
Języki programowania	x_5	0,7	1–5 pkt.
Polska wersja językowa	x_6	0,1	jest/nie ma
Oprogramowanie narzędziowe	x_7	0,7	1–5 pkt.
CASE na różnych etapach	x_8	0,6	1–4 etapy
Baza wiedzy	x_9	0,1	zawiera/nie zawiera
Zmagazynowane procedury	x_{10}	0,8	zawiera/nie zawiera
Przejrzystość dokumentacji	x_{11}	0,7	1–5 pkt.
Wspomaganie przez firmę	x_{12}	0,5	1–5 pkt.
Dynamika firmy	x_{13}	0,5	1–5 pkt.

Tabela 2. Wartości cech dla poszczególnych SZRBD

Cecha	ORACLE v. 6.0	PROGRESS v. 6	INFORMIX v. 4.0	INGRES v. 6.3	System wzorcowy
x_1	9	10	5	6	10
x_2	5	5	3	4	5
x_3	3	5	4	4	5
x_4	2	5	4	5	5
x_5	5	4	4	3	5
x_6	0	0	1	1	1
x_7	5	5	3	5	5
x_8	3	2	3	4	4
x_9	0	0	0	1	1
x_{10}	0	1	0	1	1
x_{11}	3	5	4	4	5
x_{12}	2	4	3	4	5
x_{13}	2	5	3	4	5

Tabela 3. Macierz odległości

	ORACLE v. 6.0	PROGRESS v. 6	INFORMIX v. 4.0	INGRES v. 6.3	System wzorcowy
ORACLE	X	1,329	0,883	1,336	1,427
PROGRESS		X	1,266	0,806	0,631
INFORMIX			X	1,087	1,300
INGRES				X	0,666

* * *

Analiza dendrytu wskazuje wyraźnie, że najbliżej systemu wzorcowego znalazły się SZRBD PROGRESS oraz INGRES. Tworzą one wyraźne skupienie z systemem wzorcowym. Pozostałe dwie bazy – ORACLE oraz INFORMIX są dość odległe od wzorca. Rezultatem analizy jest wniosek o dużej przydatności systemów PROGRESS i INGRES do budowy SIZ.

Należy podkreślić, że uzyskanych rezultatów analizy nie należy przyjmować w sposób uproszczony i bezkrytyczny. W każdym przypadku należy dokładnie zapoznać się z możliwościami poszczególnych systemów, a następnie wybierać do ostatecznego porównania te cechy, które są szczególnie ważne dla konkretnej aplikacji. Należy zauważyć, że w kompleksowej ocenie subiektywne jest także ustalenie wag poszczególnych cech.

LITERATURA

[1] Banaszak Z., Jampolski L.: Komputerowo wspomagane modelowanie elastycznych systemów produkcyjnych. WNT, Warszawa, 1991

dokończenie na s. 15

ZDZISŁAW SZYJEWSKI
Uniwersytet Szczeciński

Klasyfikacja narzędzi wspomagania CASE

Historia komputerowego wspomagania procesu tworzenia systemu informatycznego nie jest zbyt długa, ale można wydzielić w niej pewne okresy rozwoju. Początkowo produkowano narzędzia wspomagające wycinkowo proces wytwarzania, później występują próby wspomagania całego cyklu życia systemu. Można mówić o wycinkowych i zintegrowanych narzędziach CASE.

Inne kryterium klasyfikowania, to kryterium funkcjonalności narzędzi CASE. Mogą to być narzędzia zarówno wycinkowe, jak i zintegrowane. Dużą grupę systemów CASE stanowią narzędzia wspomagające funkcje graficzne realizowane w procesie projektowania. Należą do niej różnorodne edytory graficzne pozwalające na łatwe kreślenie, modyfikowanie i drukowanie schematów i wykresów oraz innych graficznych sposobów prezentacji działań w procesie tworzenia systemu. Edytory graficzne mogą uwzględniać specyficzną dla danej metodyki notację i symbolikę – bez wgłębiania się w badanie spójności kreślonych rysunków. Bardziej zaawansowane narzędzia CASE, oprócz właściwości edytorów graficznych, zajmują się ponadto badaniem spójności i poprawności metodycznej, której reguły są „zaszyte" w oprogramowaniu wspomagającym.

Dokumentowanie prac wykonywanych w procesie tworzenia systemu, to najczęściej wykorzystywana funkcja edytorów graficznych wspomagających proces tworzenia systemu. Łatwość utrzymania dokumentacji w stanie aktualnym, to główne zalety tej klasy narzędzi wspomagania, wykorzystywanych w procesie wytwarzania systemu.

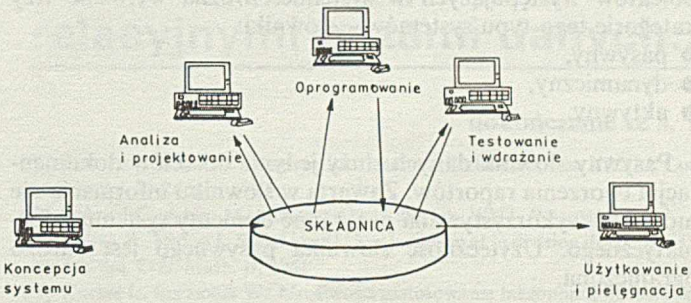
Kryteria klasyfikacyjne nie są zbyt precyzyjne i konkretne narzędzia CASE można sklasyfikować bardzo różnorodnie. Firma doradcza CASE Research opracowała obszerną i stosunkowo precyzyjną klasyfikację [3]. Według tej klasyfikacji wyróżnia się pakiety CASE:

- przechowywania danych (*repository*),
- modyfikacji, adaptacji systemów (*re-engineering*),
- wspomagania cyklu życia systemu,
- sterowania realizacją projektu,
- bieżącego doskonalenia jakości systemu.

Składnica informacji (*repository*)

Jednym z pierwszych sposobów wspomagania procesu tworzenia systemu informatycznego była idea słownika, w którym były zapisywane wszystkie definicje obiektów definiowanych w poszczególnych fazach procesu realizacji. Założeniem koncepcji słownika było zapewnienie integracji wewnętrznej oraz uproszczenie problemów dokumentacji. Najbardziej wyrafinowanym przykładem tej idei jest pakiet opisu i analizy danych PSL/PSA, opracowany na Uniwersytecie Michigan.

Składnica informacji (*repository*) jest rozwinięciem idei słownika. Rysunek obrazuje położenie składnicy w procesie tworzenia systemu. Centralne położenie wskazuje na kluczową rolę, jaką spełnia w pełnym cyklu życia systemu. Do idei słownika dodane zostały mechanizmy gwarantujące bezpieczeństwo danych, zgodność wersji oraz zarządzanie obiektami definiowanymi w systemie. Składnica jest centralną bazą danych opracowywanego systemu z własnym systemem zarządzania.



Koncepcja wytwarzania systemu informatycznego

Podstawowymi cechami składnicy informacji są:

- łatwy dostęp do definicji danych, takich jak: rekordy, struktury baz danych, formularze, procedury i reguły postępowania,
- utrzymywanie informacji o obiektach, takich jak: zbiory danych, modele, dokumenty, moduły kodu;
- utrzymanie informacji o metodach wykorzystywanych w procesie tworzenia, takich jak: operacje na obiektach, sposoby tworzenia obiektów, zasady edycji,
- dzielenie obiektów i zarządzania nimi przez zbiór dostępnych narzędzi obsługi składnicy.

Składnica informacji to podstawowy element technologii CASE wytwarzania systemu informatycznego, natomiast programy jej obsługi to narzędzia wspomagania. Narzędzia te najczęściej wywodzą się lub są wmontowane w oferowane przez różnych producentów systemy zarządzania bazą danych. Mimo różnic dzielących systemy zarządzania, charakterystycznych dla większości składnic informacji oraz obsługujących je systemów zarządzania, zadaniem tych narzędzi jest:

- powiązanie z systemem zarządzania bazą danych, z czego wynika gwarancja integralności, spójności i bezpieczeństwa danych,
- utrzymanie związków między definicjami, śledzenie i analiza wpływu wprowadzanych zmian,
- interfejs wywołań składnicy, umożliwiający komunikację z różnorodnymi środowiskami programowymi i sprzętowymi,
- rozproszenie bazy w sieci lokalnej lub rozległej,
- interakcyjny interfejs do wprowadzania zmian,
- zgodność z poprzednimi wersjami składnic,
- ukierunkowany obiektowo interfejs,

- możliwość definiowania własnych reguł, obiektów i związków między obiektami[1].

Składnica danych jest to aktywny, rozproszony słownik danych umożliwiający kompleksowe zarządzanie metainformacją całego systemu informatycznego. Wprowadza spójny, centralny opis danych wykorzystywanych w systemie przez różnorodne produkty, umożliwiając redukcje nadmiarowych i niekonsekwentnych definicji oraz zwiększenie integralności danych.

Przechowywana w składnicy metainformacja zawiera takie atrybuty, jak położenie, typ, opis, format, rozmiar i historie zmian. Każda zmiana metainformacji jest nadzorowana przez system słownika, co zapewnia wysoki stopień kontroli, zarówno nad procesami zarządzania informacją, jak i tworzonymi aplikacjami, które korzystają z definicji zawartych w składnicy.

Składnicą metainformacji był tradycyjnie kod źródłowy każdej aplikacji. Potencjalnie każdy taki kod zawierał własne definicje danych. Nadmiarowość definicji i ich niespójność były więc regułą. Zmiana definicji w dowolnym programie mogła prowadzić i najczęściej prowadziła do naruszenia integralności i spójności danych. Rozwiązaniem tego problemu jest centralne, wiarygodne i uniwersalne źródło definicji danych oraz innych obiektów występujących w systemie. Można wyróżnić trzy kategorie tego typu systemów – słowniki:
- pasywny,
- dynamiczny,
- aktywny.

Pasywny słownik danych służy jedynie do celów dokumentacji i tworzenia raportów. Zawarta w słowniku informacja nie może być wykorzystywana przez inne elementy systemu informatycznego. Użyteczność słownika pasywnego jest bardzo ograniczona.

W systemie wykorzystującym technologię dynamicznego słownika danych każda zmiana metainformacji jest natychmiast przekazywana do wykorzystujących je aplikacji. Gwarantuje to wykorzystanie zawsze aktualnej wersji definicji. Programy sięgają do słownika po potrzebne definicje podczas uruchamiania oraz każdorazowo przy dostępie do danych. Główną wadą słownika dynamicznego jest jego ograniczona wydajność. Konieczność propagowania zmian definicji do wszystkich korzystających z niej aplikacji jest bardzo czasochłonna.

Przechowywana w słowniku aktywnym metainformacja jest wykorzystywana przez programy w procesie kompilacji. Zmiany definicji nie zostają odzwierciedlone w wykonywanym module aplikacji aż do czasu kolejnej kompilacji. Operacja ta może być natomiast wykonana bez ręcznego wprowadzania zmian definicji w kodzie źródłowym. Słownik tego typu nie zmniejsza wydajności systemu, bowiem metainformacje są wykorzystywane tylko w procesie kompilacji aplikacji[2].

Najczęściej składnica oferuje zalety wszystkich wymienionych typów słowników w zależności od sposobu ich wykorzystania. Ponadto jest udostępniana architektura słownika rozproszonego, w której wiele fizycznie oddalonych słowników (istniejących na odrębnych węzłach sieci) pracuje jako jeden, spójny słownik metainformacji. Budowy składnicy wykorzystuje model obiekt-atrybut-relacja. Przez interfejs wywołań jest możliwy bezpośredni dostęp do obiektów przechowywanych w składnicy z poziomu języków programowania.

[1] Ciekawą próbę zbudowania systemu mającego takie cechy podjął zespół prof. Wojciecha Olejniczaka na Uniwersytecie Szczecińskim (por. [4]).

[2] Przykład takiego rozwiązania, zastosowany przez autora w budowie dużego systemu informatycznego, opisano w pracy [6].

Z przedstawionej charakterystyki składnicy informacji widać wyraźnie, że jest to kluczowy element technologii wytwarzania systemu informatycznego CASE. Składnica gwarantuje integrację wewnętrzną systemu, a przez mechanizmy odpowiednich interfejsów, również integrację zewnętrzną, zarówno tworzonego systemu, jak i procesu wytwarzania w pełnym cyklu życia.

Pielęgnacja systemu *(maintenance)*

Liczne badania wskazują, że duża część kosztów wytwarzania systemu informatycznego jest ponoszona na obsługę zmian i modyfikacji, jakie powstają w trakcie tworzenia systemu, jak również w trakcie eksploatacji użytkowej. Zmiany i modyfikacje sprowadzane do systemu stanowią źródło licznych komplikacji w procesie wytwarzania. Wspomaganie tych prac jest uzależnione od technologii wytwarzania.

Istotą prawidłowego modyfikowania systemu jest prawidłowe przewidywanie wpływu wprowadzonych zmian na inne elementy systemu, jak również konieczność niezbędnych zmian w innych fazach projektu. Wyróżnia się trzy podstawowe sytuacje:
- restrukturyzacja systemu (ang. *restructuring*),
- modyfikacja wstępująca (ang. *rewerse engineering*),
- modyfikacja zstępująca (ang. *forward engineering*).

Całkowita przebudowa projektu systemu może wynikać z potrzeby dostosowania systemu do istotnych zmian, jakie zaszły w obiekcie, gdzie jest stosowany system informatyczny. Istotne jest, aby restrukturyzację wykonać możliwie szybko, wykorzystując maksymalnie wcześniej opracowane elementy. Najczęściej wymaga to jednak przejścia przez cały cykl życia systemu.

Modyfikacje zstępujące to sytuacje, w których zmiany we wstępnych fazach cyklu życia systemu implikują odpowiednie modyfikacje w dalszych fazach tego cyklu. Odwrotnie dzieje się w przypadku modyfikacji wstępującej, gdzie zmiany są inicjowane w fazie wdrożenia lub programowania i specyfikację należy odpowiednio zmodyfikować[3].

Wychodząc naprzeciw stanowi faktycznemu, polegającemu na tym, że brak jest najczęściej aktualnej dokumentacji użytkowanych systemów, rozwijane są narzędzia, których podstawowym celem jest wytwarzanie specyfikacji na podstawie użytkowanego produktu. Działania takie (*re-engineering*) są niezbędne wtedy, gdy jest konieczna modyfikacja lub restrukturyzacja systemu nieodpowiednio udokumentowanego lub tworzonego w innej technologii. Pozwala to na ochronę poczynionych wcześniej inwestycji na prace projektowe, przez możliwość powtórnego przekształcenia i modyfikacji projektu. W pracy [2] są opisane produkty firmy Bachman, wchodzące w skład koncepcji IBM AD/Cycle, realizujące takie procedury.

Wspomaganie cyklu życia systemu

W literaturze można spotkać wiele różnych modeli cyklu życia systemu. Różnice między poszczególnymi modelami z reguły nie są istotne i sprowadzają się do nazewnictwa lub różnic w podziale na fazy, etapy realizacji w takiej lub innej konfiguracji. Jednym z takich modeli jest wspomniana metodyka punktów węzłowych.

Na potrzeby wspomagania metodyki punktów węzłowych został stworzony prosty pakiet komputerowego wspomagania

[3] W dziełaniach tych bardzo pomocna może być sieć zależności między punktami węzłowymi, opisana w ramach metodyki punktów węzłowych (por. [7]).

EKSPERT [7]. Założenia oraz zasady metodyki zostały zaszyte w rozwiązaniach programowych pakietu programowego. Użytkownik pakietu ma możliwość twórczego rozwinięcia metodyki, poczynając od pewnego poziomu ogólności rozwiązań, zgodnie ze specyfiką tworzenia konkretnego systemu informatycznego. Pakiet jest ukierunkowany głównie na:

- kontrolę merytorycznej poprawności realizacji procesu tworzenia systemu,
- wspomaganie tworzenia i utrzymania w stanie aktualnym dokumentacji systemu[4].

Wbudowana w pakiet programowy metodyka punktów węzłowych obejmuje pełny cykl życia systemu. W części objętej wspomaganiem komputerowym pakiet koncentruje uwagę na tym, jakie czynności w kolejnych fazach realizacji systemu informatycznego należy wykonać oraz w jakim porządku. W pakiecie wbudowano sieć zależności punktów węzłowych, co umożliwia kontrolę poprawności logicznej realizacji poszczególnych czynności oraz wyświetlanie na ekranie czynności, jakie można rozpocząć realizować po zakończeniu prac nad danym punktem węzłowym.

Pakiet jest przewodnikiem w realizacji systemu, nie ograniczającym inwencji twórców, gdyż oprócz wbudowanych powiązań między punktami węzłowymi, istnieje możliwość własnego kreowania czynności, poczynając od określonego poziomu ogólności punktów węzłowych. Użytkownik na podstawie własnej wiedzy i doświadczenia, może określić skład czynności wchodzących w zakres punktu węzłowego lub skorzystać z wiedzy zawartej w pakiecie. Własny opis zakresu tematycznego poszczególnych punktów węzłowych metodyki daje możliwość twórczej modyfikacji szkieletu metodyki, wbudowanego w pakiecie, oraz adaptacji tej metodyki do konkretnych potrzeb tworzonego systemu informatycznego.

Oprócz funkcji wspomagania kontroli procesu realizacji, pakiet EKSPERT wspomaga tworzenie dokumentacji systemu, dostarczając wygodne mechanizmy edytorów tekstu w połączeniu ze standardowym spisem treści, wywodzącym się z listy punktów węzłowych [5].

Narzędzia wspomagające poszczególne fazy realizacji systemu informatycznego są klasyfikowane w literaturze w zależności od tego, który etap prac wspomagają. Występuje tu podział na:

- *front-end* CASE, narzędzia wspomagające fazę planowania, analizy i projektu,
- *back-end* CASE, narzędzia wspomagające fazę konstrukcji systemu, czyli jego projekt i oprogramowanie.

Inny podział to:

- *upper*-CASE,
- *middle*-CASE,
- *lower*-CASE,

dzielący narzędzia komputerowego wspomagania w zależności od tego, jaką fazę cyklu życia systemu wspomaga. Narzędzia klasy *lower* najczęściej wspomagają fazę kodowania, testowania i prototypowania, kompilacji programów oraz wdrożenia. Klasa *upper*, to narzędzia wspomagające fazę planowania, analizy i specyfikacji[5].

[4] Pakiet programowy EKSPERT autorstwa Zdzisława Szyjewskiego i Ireneusza Szydłowskiego wraz z niezbędną dokumentacją, jest produktem handlowym.

[5] Jeszcze inne klasyfikacje oraz przykłady produktów, w tym krajowej produkcji, reprezentantów poszczególnych klas można znaleźć w pracy [1].

LITERATURA

[1] Fuglewicz P.: Systemy CASE – problemy, techniki, rozwiązania. W: „Komputerowe wspomaganie tworzenia systemów informatycznych" – materiały Trzeciej Wiosennej Szkoły PTI, Świnoujście 1990

[2] Madrjas K.: AD/Cycle. IBM-owska filozofia inżynierii projektowania oprogramowania. W: „Metodologie tworzenia systemów informatycznych a CASE" – materiały Piątej Wiosennej Szkoły PTI, Świnoujście 1992

[3] Marlyn V., Boone G.: Case Product Classification Model. CASE Bulletin, March 1989

[4] Olejniczak W., Szydłowski I.: FACTOR – system wspomagania projektowania systemów informatycznych. W: „Komputerowe wspomaganie tworzenia systemów informatycznych" – materiały Trzeciej Wiosennej Szkoły PTI, Świnoujście 1990

[5] Szyjewski Z.: Komputerowe wspomaganie dokumentowania tworzenia systemów informatycznych. W: „Komputerowe wspomaganie tworzenia systemów informatycznych" – materiały Trzeciej Wiosennej Szkoły PTI, Świnoujście 1990

[6] Szyjewski Z.: Metoda projektowania i realizacji oprogramowania złożonego systemu informatycznego na przykładzie systemu TRANSTER. W: „Informatyka w zarządzaniu przedsiębiorstwem" mat. konf. INFROGRYF'84, TNOiK Szczecin 1984

[7] Szyjewski Z.: Projektowanie systemów informatycznych zarządzania. W: „Metodyki projektowania systemów informatycznych" – mat. konf. Pierwszej Wiosennej Szkoły PTI, Świnoujście 1988.

Systemy zarządzania relacyjnymi bazami danych...

dokończenie ze s. 12

[2] Butler M., Bloor R.: Comparative Database Performance on VAX. A Computer Research Report, 1990

[3] Delobel C., Adiba M.: Relacyjne bazy danych. WNT, Warszawa, 1989

[4] Florek K., Perkal J., Steinhaus H.: Sur la liaison at la division des points d'un ensamle fini. Coll. Math. II, 1951

[5] Korbel L., Szymonik W.: Możliwości zastosowania taksonomii wrocławskiej i obszarowej dla porównywania zdolności produkcyjnej obiektów gospodarczych. Wyd. Politechniki Lubelskiej, Ekonomika nr 12, 1981

[6] Materiały Drugiego Forum Technologii Informatycznych PTI. Mrągowo, 1992

[7] Metodologie tworzenia systemów informatycznych a CASE. Materiały 5. Wiosennej Szkoły PTI, Świnoujście, 1992

[8] Systemy relacyjnych baz danych: INFORMIX, INGRES, ORACLE, PROGRESS. Materiały trzeciej i czwartej Górskiej Szkoły PTI, Szczyrk, 1992

[9] Współczesne obszary zastosowań informatyki. Materiały IV Wyższej Górskiej Szkoły PTI, Szczyrk, 1992.

W związku z wprowadzeniem powszechnego podatku dochodowego i koniecznością informowania właściwych Urzędów Skarbowych o wszystkich wypłatach zwracamy się do Autorów wysyłających teksty do opublikowania o podawanie następujących danych:

- **nazwisko, imiona (pierwsze i drugie),**
- **imiona: ojca i matki,**
- **miejsce i data urodzenia,**
- **numer identyfikacyjny PESEL (wpisany przez Biuro Meldunkowe do dowodu osobistego – nie mylić z numerem dowodu osobistego!),**
- **dokładny adres (miejsce zameldowania),**
- **adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora (bardzo ważne – bez tej informacji nie możemy przekazać honorarium do wypłaty!).**

INFORMATYKI nie kupisz w kiosku, ale możesz ją już zaprenumerować na poczcie.

HENRYK SROKA
Akademia Ekonomiczna
Katowice

Systemy wspomagania decyzji z bazą wiedzy

Proces transformacji polskiej gospodarki z centralnie planowanej do rynkowej obejmuje nie tylko typowe organizacje przemysłowe, lecz także liczne inne instytucje tj. banki, handel, small business, transport itp. Powoduje on dynamiczne przyspieszenie zmian w zarządzaniu gospodarką narodową, organizacjami oraz w procesach podejmowania decyzji kierowniczych. W tej sytuacji pojawiają się realne potrzeby intensywnego włączenia wspomagania komputerowego do tworzenia lub przekształcania systemów informacyjnych w nowej rzeczywistości organizacyjnej.

Opracowanie opiera się na własnych pracach badawczych, projektowych i wdrożeniowych prowadzonych w latach 1980–1992 w kraju, jak również na szerokiej wymianie doświadczeń z ośrodkami naukowymi za granicą (USA, Japonia, Europa Zachodnia [4]). Zaprojektowane i wdrożone systemy informatycznej typu SWD (system wspomagania decyzji), SWD-BW (system wspomagania decyzji z bazą wiedzy) oraz SE (systemy ekspertowe jako element SWD) obejmują różne etapy procesu transformacji polskiej gospodarki w latach 1986–1992 [6]. Liczne projekty wykonano w celu lepszego zrozumienia potrzeb informacyjnych oraz zakresu niezbędnej wiedzy w różnorodnych procesach podejmowania decyzji na różnych etapach transformacji organizacji [7].

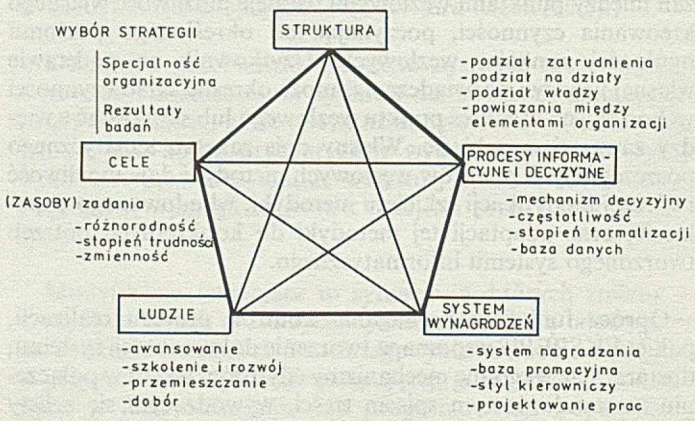
Systemy te projektowano również z zamiarem zbadania możliwości oferowanych przez różne metodologie budowy SWD (SWD ukierunkowane na modele, SWD oparte na generatorach, SWD z bazą wiedzy, SWD z SE) dla doskonalenia podejmowania decyzji organizacyjnych w dynamicznym otoczeniu. Wspomniane badania i system realizowano głównie w ramach projektów finansowanych przez instytucje rządowe w latach 1980–1990 oraz 1991–1992 [5].

Ogólne ukierunkowania budowy SWD

W organizacji wytwórczej, która jest podstawowym obiektem gospodarki, występuje wiele problemów decyzyjnych słabo ustrukturalizowanych lub nieustrukturalizowanych, złożonych i nie powtarzających się. Ten rodzaj problemów decyzyjnych związany jest z wyznaczeniem strategii działania i obejmuje decydentów szczebla naczelnego lub ich bezpośrednich współpracowników.

Celem realizacji SWD jest konstrukcja narzędzia informatycznego dla decydenta. Mamy więc do czynienia z decydentem – osobą i wszystkimi psychologicznymi właściwościami myślenia ludzkiego i to bardzo specyficznego rodzaju myślenia. Decydent funkcjonuje w organizacji, co wywołuje potrzebę uwzględnienia wszystkich uwarunkowań towarzyszących czynnościom decydenta w interakcji z organizacją (przełożonymi, podwładnymi, osobami na tym samym poziomie zarządzania, osobami z zewnątrz, grupami osób itp.).

Skoro zamierza się wkraczać z komputerem i informatyką do przedsiębiorstwa w celu usprawnienia jego działania przez procesy informacyjne, należy wcześniej uświadomić sobie, czym jest przedsiębiorstwo, jaką rolę pełni w nim decydent i czy w ogóle można projektować organizację przedsiębiorstwa [2].



Rys. 1. Koncepcja projektowania organizacji

Rozwój dotychczasowej wiedzy i praktyki organizacji oraz zarządzania wskazuje na to, że aby projektować lub doskonalić przedsiębiorstwo, należy brać pod uwagę pięć zbiorów zmiennych opisujących jego działanie (rys. 1) w postaci:

- celów i zadań,
- procesów informacyjno-decyzyjnych,
- struktury,
- systemu motywacyjnego,
- zatrudnionych osób.

Jeśli zamierza się doskonalić zarządzanie przedsiębiorstwem przez doskonalenie procesów informacyjno-decyzyjnych opierając się na informatyce, to należy przewidywać stosowną interakcję wprowadzanych udoskonaleń w stosunku do pozostałych zmiennych opisujących przedsiębiorstwo.

Po drugie, zarówno przedstawiciele nauki, jak i praktyki poświęcają w ostatnich latach szczególnie dużo uwagi badaniom procesów podejmowania decyzji i decydentom. Nadal jednak występuje ogromna luka między wiedzą i praktyką o procesach podejmowania decyzji. Znani autorzy nie są zgodni w określeniu tego jakie czynności wykonuje decydent i w jaki sposób je realizuje [8]. Nie ulega jednak wątpliwości, że decydenci są specjalnego rodzaju ludźmi, obarczonymi wyróżnioną funkcją do przewodzenia, wyzwalania i łączenia utajonych zdolności oraz marzeń innych ludzi. Czynności kierownicze mają charakter najbardziej rozległy, najbardziej potrzebny,

nieporównanie do innych złożony – najbardziej subtelny z ludzkich działań. Można dalej stwierdzić, że podejmowanie decyzji jest najistotniejszym z zadań kierowniczych, które przenika wszystkie inne funkcje i role kierownicze. Polega to przede wszystkim na [9]:

- wyborze całościowej strategii działania,
- ustaleniu zadań specjalnych,
- delegowaniu odpowiedzialności,
- ocenie wyników,
- inicjowaniu zmian.

Podejmowanie decyzji jest więc procesem sekwencyjnym, a nie jednorazowym czynem zrealizowanym raz na zawsze. Na decyzję aktualną oddziaływuje bowiem decyzja poprzednio podjęta i w ten sposób proces jest rozwijany w czasie. W tym charakterze jest to proces polegający na rozpoznaniu i badaniu pojawiającego się problemu, sformułowaniu i ocenie wariantów jego rozwiązania, wyboru kierunków działania aż do jego wdrożenia.

Po trzecie, dynamicznie rozwijające się zastosowanie komputera w zarządzaniu przedsiębiorstwem było spowodowane nie tylko rozwojem techniki komputerowej, ale również wszystkich dyscyplin naukowych biorących udział w kreowaniu ruchu interdyscyplinarnego, tj. teorii organizacji i zarządzania, psychologii, badań operacyjnych, cybernetyki itp.

Struktura SWD

W końcu lat siedemdziesiątych na problem klasyfikacji systemów wspomagania decyzji można było spojrzeć co najmniej z dwóch punktów widzenia: stopnia złożoności działań, które wspomaga system, lub zakres, w jakim oddziałuje on na podejmowaną decyzję (tabela).

Początkowa klasyfikacja SWD

Operacje podstawowe w podejmowaniu decyzji	Rodzaje SWD
Wyszukiwanie pozycji	System gromadzenia zbiorów
Dostarczanie raportów	System analizy danych
Formułowanie problemów lub rozwiązań	System analizy i informacji
Ocena skutków wariantów	Modele ekonomiczne
Pozycje wariantów	Modele opisowe
	Modele optymalizacyjne
Podejmowanie decyzji	Modele sugerujące

W przypadku pierwszym otrzymujemy siedem rodzajów SWD od najprostszych (gromadzenia zbiorów) do modeli sugerujących rozwiązania [11]. W drugim przypadku można wyróżnić sześć rodzajów systemów – od wyszukiwania pozycji do wyboru najkorzystniejszego rozwiązania. Między omawianymi klasyfikacjami istnieje wiele podobieństw. Klasyfikacje te można uprościć do grup systemów ukierunkowanych na dane lub na modele.

Badania cech osobowościowych decydenta oraz niedomagania dotychczasowych propozycji rozwiązań SWD wskazują na potrzebę ukierunkowania rodzajów wspomagania na:

- wspomaganie wyobraźni i poszukiwania pomysłów w procesie pdejmowania decyzji za pomocą łatwych „obrazów", takich jak rysunki, schematy, wykresy itp.,
- pomocy przy przypominaniu wiedzy zawartej w pamięci decydenta,
- wspomagania typowych czynności występujących w procesie podejmowania decyzji (ustalenia problemu, projektowania rozwiązań, wyboru, wdrażania, oceny),
- wspomagania indywidualnych sposobów postępowania decydenta w procesie podejmowania decyzji, wypływających z reprezentowanego stylu, umiejętności i wiedzy,
- pomoc przy kontroli dostarczanego wspomagania, w celu upewnienia się decydenta, że wszystko w procesie podejmowania decyzji przebiega zgodnie z jego osobistymi odczuciami [10].

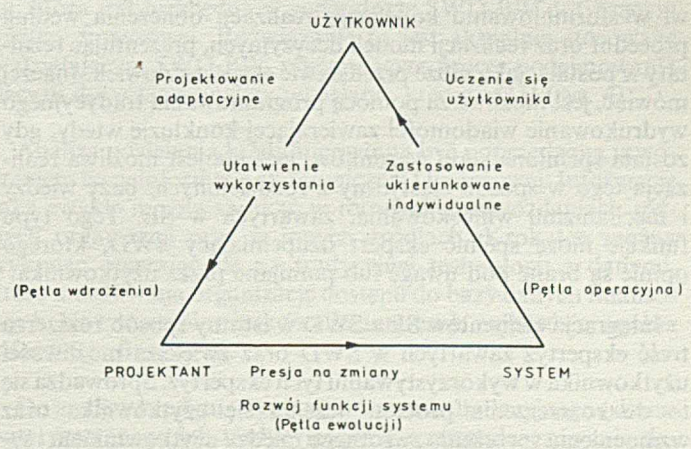
Proces projektowania wcześniejszych systemów informatycznych zakładał, że albo struktura problemu decyzyjnego jest znana, albo jest poznawalna na drodze interakcyjnej. Oznaczało to, że ustalenie problemu, preferencje kierownicze, procedury podejmowania decyzji, otoczenie decydenta itp. są niezmienne. Przy projektowaniu SWD mamy do czynienia z tzw. skutkiem migracji w umyśle użytkownika i projektanta, wywołanym niestrukturalizowanym lub częściowo zestrukturalizowanym procesem budowy systemu. Skuteczne projektowanie SWD wymaga więc integracji wysiłków co najmniej dwóch „aktorów":

- decydenta lub użytkownika, który współpracuje przy budowie i wykorzystuje system,
- projektanta, który opracowuje szczegółową strukturę i możliwości systemu oraz kieruje jego wdrożeniem.

Trudno już dzisiaj zaproponować uniwersalną metodologię projektowania SWD. Nagromadzone dotychczas doświadczenia umożliwiają określenie wielu zleceń, na których w zależności od konkretnych potrzeb, może opierać się projektant przy formułowaniu własnej koncepcji SWD.

Można wymienić cztery właściwości związane z projektowaniem SWD:

- uczenie się decydenta i potrzeba dynamicznej zmiany systemu w istotny sposób utrudniają opracowanie dokładnych założeń systemu;
- proces projektowania ma charakter ewolucyjny, decydent zmierza do coraz doskonalszej postaci systemu;
- czynności początkowe, kiedy decydent może włożyć więcej wysiłku i wydać więcej pieniędzy, przechodzą w fazie późniejszej w bardziej usztywnione oraz ukierunkowane;
- można wyróżnić w procesie projektowania dwa podejścia. Pierwsze – „słabe", zmierzające do uwzględnienia preferencji użytkownika przy istniejących możliwościach komputera. Drugie – „silne", związane z presją na użytkownika ze strony zleceniodawcy lub projektanta, kiedy użytkownik zachowuje się pasywnie i obawia się zmian.



Rys. 2. Zarys adaptacyjnego procesu budowy SWD

Na strategię budowy SWD ma oczywiście wpływ wybór dokonywany między zakresem a głębokością wspomagania procesu decyzyjnego. Głębokość wiąże się z tym, jak obszernie za pomocą komputera zamierzamy rozwiązywać poszczególną fazę, np. ustalenie alternatyw. Zakres wskazuje, ile faz procesu

decyzyjnego zamierzamy wspomagać. Kluczem do określenia właściwej strategii budowy SWD jest zapewne potrzeba „uczenia się" decydenta. SWD nie rozwiązuje problemu do końca, lecz zachęca do penetracji i poszerzenia własnych umiejętności w jego rozwiązywaniu. Wynika stąd postulat budowy na początku prostego wspomagania i następnie jego rozbudowy. System inicjujący musi być na tyle bliski rozwiązaniu rzeczywistemu, aby stać się łatwym i atrakcyjnym w użyciu. Ten rodzaj podejścia będziemy nazywać strategią adaptacyjną. Skuteczny proces budowy SWD zależy od ciągłej i harmonijnej interakcji użytkownika, projektanta i systemu (ryś. 2).

Systemy wspomagania decyzji z bazą wiedzy

Jedna z końcowych konkluzji wcześniejszych rozważań prowadzi do stwierdzenia, że metodologia systemów ekspertowych wnosi dwie nowe możliwości:
- budowę systemu, który może naśladować rozumowanie,
- budowę systemu, który może opisać przebieg tego rozumowania oraz uzasadnienie wniosków.

Wydaje się oczywiste, że właściwości te mogą być bardzo pomocne przy budowie SWD. Nasze prace w zakresie integracji wiedzy SE i metodologii budowy SWD rozpoczęły się w 1986 r.

Rozszerzenie możliwości SWD przez wykorzystanie SE

Klasyczny już dzisiaj model SWD możemy rozszerzyć w sześciu różnych kierunkach [3].

Porada ekspercka w specyficznym obszarze problemów

Na tym poziomie dostarczenie ekspertyzy wybiega poza możliwości SWD i należałoby poprosić o poradę eksperta. Zwiększenie możliwości SWD w tym kierunku wymaga bazy wiedzy, np. obejmującej analizę finansową dla SWD wspomagającego organizację w zakresie decyzji kredytowych.

Jednym z fundamentalnych aspektów porady eksperckiej jest potrzeba dysponowania znaczną ilością symbolicznego rozumowania jako nową przestrzenią w funkcjach SWD.

Rzecz polega na tym, że obecny SWD pomaga użytkownikowi w sformułowaniu koncepcji, realizacji obliczenia według procedur oraz realizacji modeli decyzyjnych, prezentując rezultaty w postaci, jaką może przedstawić ekspert-człowiek. Inaczej mówiąc, jest możliwe za pomocą programowania tradycyjnego wydrukowanie wiadomości zawierającej konkluzję wtedy, gdy została spełniona seria warunków, lecz nie jest możliwa realizacja tego w sposób elastyczny z rozdzielonych: bazy wiedzy i mechanizmu wnioskowania, zawartych w SE. Tego typu funkcje może spełnić ekspert uzupełniający SWD, którego opinie są brane pod uwagę lub pomijane przez użytkownika.

Integracja elementów SE z SWD w istotny sposób rozszerza treść ekspertyz zawartych w SWD oraz zwiększa możliwości użytkownika w wykorzystywaniu tych ekspertyz. Sprowadza się to do rozszerzenia procesu uczenia się użytkownika oraz wzmocnienia sprzężenia zwrotnego między użytkownikiem i systemem.

Wyjaśnienie wniosków eksperta

Jak wcześniej wspomniano, dodatkowa wartość SE polega na rozszerzeniu możliwości systemu o wyjaśnienie przeprowadzonego procesu rozumowania. Możliwość ta jest niezwykle istotna, ponieważ wypełnia często formułowaną w badaniach ideę, że dobry SWD winien udoskonalić proces uczenia się użytkownika. Funkcja objaśniania jest istotna co najmniej z kilku powodów np.:
- użytkownik intensywnie poszukuje lepszego rozwiązania i uzyskuje większe zaufanie do systemu,
- założenia zawarte w systemie powstają raczej explicite zamiast stałych implicite.

Inteligentne wspomaganie analizy decyzji

Można dostrzec, że inteligentne wspomaganie analizy decyzji jest jedną z najistotniejszych konsekwencji integracji SWD i SE. Pierwszym krokiem do zrozumienia modelu jest zdolność czytania tekstu modelu (zwykle zbiory równań) i łatwego rozumienia struktury logicznej (strukturalnej zależności między zmiennymi). Jest również oczywiste, że korzyści języka modelowania w stosunku do arkusza kalkulacyjnego polegają na możliwości klarownego wyrażania nie tylko związków w modelach decyzyjnych, lecz także ogólnych struktur modeli. Jedną z niezwykle użytecznych możliwości SWD jest możliwość znalezienia i wyświetlenia definicji zmiennej modelu.

Jak powszechnie wiadomo, wielu decydentów nie stosuje metod sformalizowanych ponieważ nie wiedzą, jak je wykorzystać. Na przykład, wielu decydentów nie pamięta przyjętych założeń dla niektórych zastosowań modelu regresji wielokrotnej. Użyteczność skorzystania z eksperta jest w wielu wypadkach oczywista, lecz w wielu przypadkach koszt lub nieosiągalność takiej możliwości eliminują zastosowania wspomnianych metod.

Znacznie poprawia tę sytuację wbudowanie elementu ekspertowego, zawierającego bazę wiedzy ze statystyki, prognozowania krótkookresowego i wybranych metod badań operacyjnych.

Inteligentna pomoc informatyczna ma wtedy trzy cele:
- prowadzenie użytkownika-nowicjusza, aby zastosował metodę w sposób właściwy,
- pomóc użytkownikowi uczenia się właściwych strategii wykorzystania narzędzi (rozwinięcie heurystyki),
- wykrycie obszaru wiedzy z dużej bazy danych.

Pomoc w wykorzystaniu narzędzi SWD – rozwój inteligentnego dialogu z użytkownikiem

SWD są zwykle wyposażone w następujące narzędzia:
- bazy danych i zbiory danych,
- modele decyzyjne,
- sprawozdanie i obrazy graficzne,
- bazy wiedzy.

Pomoc przy formułowaniu pytań

W przypadku dużej bazy danych ze złożonymi zależnościami, formułowanie wielu pytań może być trudne. SE może dopomóc użytkownikowi przy formułowaniu jego pytań.

Inteligentne wspomaganie w procesie budowy modelu

Idea inteligentnego wspomagania budowy modeli sprowadza się do tego, że wiedza określonej dziedziny może być wykorzystana do budowy modeli analitycznych. Dotyczy to oczywiście takich problemów, które można sformalizować za pomocą narzędzi modelowania. W literaturze idea ta jest często opisywana pod nazwą zarządzania modelami. Obejmuje to wiedzę, jaka może być pomocna użytkownikom w procesie formułowania problemu i polega na dostarczaniu (wskazywaniu) właściwych dla problemu bloków modeli.

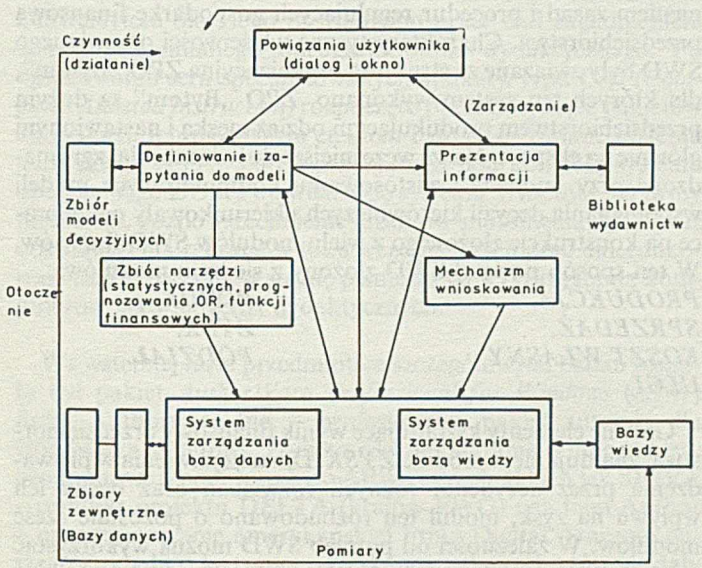
Z powyższych przykładów wynika, że celem SWD-BW jest integracja tradycyjnego SWD (zarządzania danymi i językiem modelowania, metodologii decyzji, wyświetlania danych numerycznych itd.) z nowymi możliwościami SE, tj. rozumowaniem symbolicznym oraz możliwościami objaśniania.

W naszych rozważaniach pragniemy pozostać w ramach paradygmatu SWD, tj. przy wspomaganiu podejmowania decyzji. Podczas rozwiązywania problemów decyzyjnych, pragniemy zająć się bardzo specyficznymi problemami, wymagającymi dostarczenia dla ich rozwiązania ekspertyzy. Ekspertyza ta mogłaby być dostarczana z bazy wiedzy oraz możliwości rozumowania.

Integracja SWD i SE prowadzi do nowej koncepcji systemu, którą będziemy nazywać SWD-BW. Do podstawowych dyscyplin, które stanowią fundament tej koncepcji, należą: lingwistyka, logika formalna, psychologia, nauki informatyczne, matematyka, statystyka oraz ekonomia. Wspomniane dyscypliny ułatwiły rozwój teorii, metod oraz technik, takich jak sztuczna inteligencja, systemy informatyczne zarządzania, behawioralne podejmowanie decyzji, analizę decyzji i badania operacyjne. Zastosowanie tych metod i technik w obszarze podejmowania decyzji doprowadziło do ewolucji dwóch rodzajów systemów dla wspomagania decyzji SWD i SE.

Zarys nowej koncepcji

Strukturę SWD-BW definiujemy przez podsystemy wraz z istniejącą między nimi integracją (komunikacja i sterowanie), jak również z przyjętą hierarchizacją.



Rys. 3. Struktura funkcjonalna narzędzi budowy SWD-BW

Zarys koncepcji podano na rys. 3. System składa się z następujących podsystemów:

- zarządzania danymi lub zarządzania bazą danych,
- wyświetlania,
- modelowania (system i użytkownik),
- statystyki i optymalizacji,
- dialogu z użytkownikiem,
- mechanizmu wnioskowania,
- zarządzania bazą wiedzy.

Łatwo można dostrzec różnice między podstawowymi i nowymi elementami SWD (modelowanie, wyświetlanie, moduły narzędzi: algorytmy statystyczne, optymalizacyjne i prognozowania). Nowymi elementami są tutaj mechanizm wnioskowania oraz system zarządzania bazą wiedzy.

Mechanizm wnioskowania został wydzielony w postaci odrębnego podsystemu, zdolnego do realizacji wielu funkcji SE w SWD-BW. W przypadku analizy finansowej przedsiębiorstwa, różne BW można wykorzystać do formułowania diagnoz finansowych, jak również strategii finansowej.

Przykład niezbędnej integracji między elementami SWD-BW

Integracja jest podstawowym problemem, z którym spotyka się projektant opracowujący narzędzia SWD-BW. Jak już wcześniej dostrzegliśmy, występuje istotna różnica między kompatybilnością dwóch programów (systemów lub podsystemów), a integracją między dwoma programami (systemów i podsystemów).

Podejmowano liczne przedsięwzięcia w zakresie powiązania przez wspólne zbiory istniejących systemów zarządzania bazą danych, bazą modeli i systemów ekspertowych. Wykonano wiele powiązań w celu łączenia arkuszy kalkulacyjnych z oprogramowaniem SE, np. w SWD-BW analizy finansowej. Niestety, większość tych wysiłków zakończyło się niepowodzeniem, ponieważ zbiór powiązań mógł być tylko wtedy użyteczny, jeśli informacja potrzebna w systemie ekspertowym jest znana wcześniej, co niestety zdarza się niezwykle rzadko.

Informacje potrzebne w SE ulegają zmianie wraz z dostępnością danych i interakcją z użytkownikiem. W konsekwencji działania SE, przemianom ulega BW, jak również modyfikacjom ulega zawartość BD.

To, co rzeczywiście jest tutaj potrzebne, to budowa generatora bazy wiedzy SWD, który należy zaprojektować od początku, między modułem systemu ekspertowego i tradycyjnymi modułami SWD.

Przykłady SWD-BW

Powyższe rozważania ilustrują następujące trzy przykłady systemów.

System wspomagania decyzji przy udzielaniu kredytu

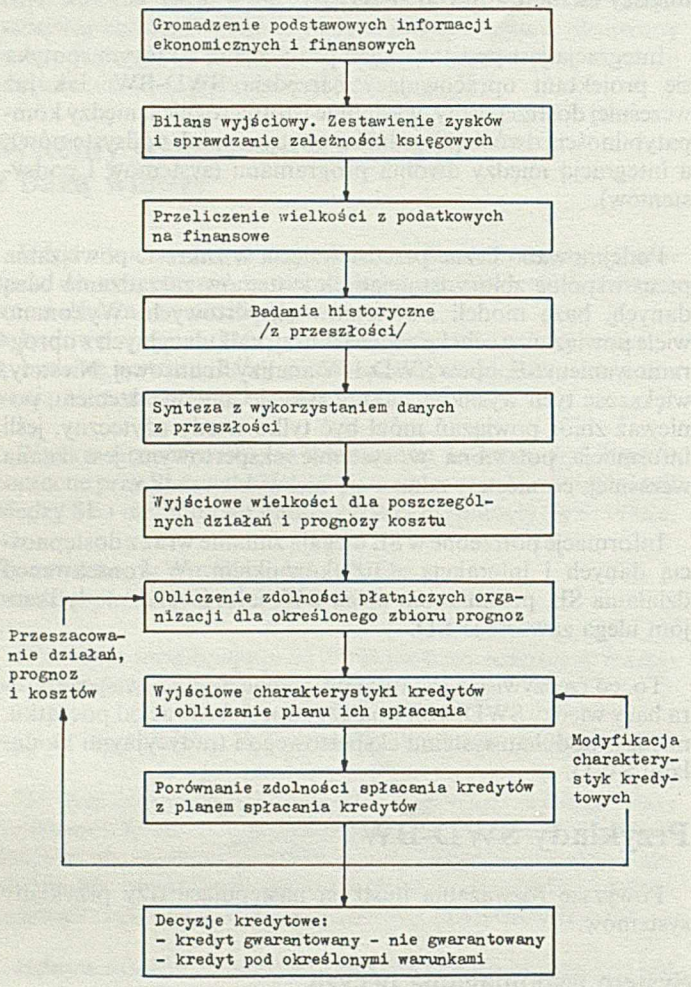
Poniżej zostanie scharakteryzowany prosty przykład interakcji, która jest niezbędna w generatorze SWD-BW dotyczącym analizy finansowej. Rozwiązanie to jest aktualnie stosowane w działającym SWD-BW, obejmującym proces podejmowania decyzji dotyczącej analizy udzielania kredytu [11] (rys. 4).

Analiza udzielania kredytu powinna być poprzedzona utworzeniem bazy informacji ekonomicznej i finansowej. Informację taką zwykle można spotkać w rocznych sprawozdaniach różnych organizacji oraz w innych źródłach. Krok ten można wykonać opierając się na tradycyjnej technologii z danymi, które umożliwiają organizację dostępu do bazy danych finansowych. Na tym etapie użytkownika interesuje najbardziej ogólna informacja finansowa.

Do analizy kredytu używa się arkusza kalkulacyjnego (skomputeryzowanego lub tradycyjnego) dla zestandaryzowania danych i sprawdzenia podstawowych zależności ekonomicznych. W tym przypadku mechanizm wnioskowania z bazą wiedzy obejmujący rachunkowość może dopomóc przy sprawdzaniu spójności w rozliczeniach rachunkowości.

Podstawowe informacje, to zwykle informacje podatkowe, gromadzone i przetwarzane przez analityka na stosowne rapor-

ty i analizy finansowe. Ten krok może być wspomagany przez zawarte w generatorze SWD-BW modelowanie danych oraz element raportów. Do analizy potrzebne jest zwykle określenie wskaźników finansowych (przepływu gotówki, kapitału obrotowego, potrzeb w zakresie kapitału obrotowego) oraz wyświetlenie lub wydrukowanie różnych form wyników (sprawozdań i wykresów).



Rys. 4. Proces decyzyjny zastosowany przez analityka kredytów podczas udzielania kredytu przedsiębiorstwu

Ten rodzaj pracy może być wspomagany przez generowanie sprawozdań i wykresów przez generator SWD-BW. Kilka raportów powinno być przeznaczonych dla dyrekcji. Wśród typowych raportów proponuje się zwykle takie, jak zestawienia zysków i bilansu, przepływu funduszy oraz zagregowanych wskaźników finansowych. Istotnym elementem wspomagania są również różnorodne prezentacje graficzne. Na tym etapie moduł EKSPERT jest potrzebny do tego, aby sprawdzić wnioski, wykorzystując informacje z kroku trzeciego i czwartego.

Jeśli kredyt jest związany z ryzykiem, to decyzji nie można podjąć jedynie na podstawie danych z przeszłości i należy wtedy rozważyć również możliwości spłaty w przyszłości. Dlatego analityk powiniem ustalić wielkości poziomu produkcji oraz przyszłych kosztów i na tej podstawie wyliczyć przyszły przepływ gotówki.

Ten etap można wspomagać za pomocą modułu systemu zarządzania oknem i funkcjami graficznymi zawartymi w generatorze SWD-BW. Należy dodać, że kontrola sprawdzania rozumowania wstecznego jest przekazywana z mechanizmu wnioskowania do modelu.

Po podaniu charakterystyk kredytu (sumy, okresu, oprocentowania i początku okresu spłacania) można przystąpić do przygotowania harmonogramu spłat. Należy pamiętać o skutkach zaciągnięcia kredytu w postaci zmian z zysku, modyfikacji struktury finansowej, przepływie funduszy itp.

Przedsiębiorstwo winno mieć w przyszłości dostateczny przepływ gotówki aby zapewnić spłacanie kredytu. Rzeczywistość przynosi jednak wiele niepewności (możliwości spłacania kredytu), wobec czego analityk może dołączyć wyliczenie dotyczące marginesu bezpieczeństwa oraz oszacowania ryzyka przy stosowaniu innych danych.

Na podstawie analizy danych z przeszłości oraz analizy informacji uzyskanych z badania prognoz, analityk może dokonać powtórnego oszacowania lub założeń wyboru decyzji.

System wspomagania decyzji

Zadaniem SWD w obszarze finansów jest ułatwienie decydentowi podejmowania decyzji w zakresie tworzenia i podziału wyniku finansowego przedsiębiorstwa. System ma pomóc w identyfikacji różnorodnych uwarunkowań sytuacji decyzyjnej, uświadomieniu różnych możliwości rozwiązania problemu, wyborze rozwiązania i w rezultacie podjęcia decyzji znajdującej się w obszarze działania decydenta.

Interesującą zaletą SWD jest umożliwienie śledzenia (symulacji) skutków różnowariantowych decyzji i w ten sposób oceniania ich przydatności gospodarczej. Istotnym atrybutem systemu jest wielokrotne przyspieszenie obliczeń.

System wspomagania decyzji był konstruowany z uwzględnieniem zasad i procedur regulujących gospodarkę finansową przedsiębiorstwa. Charakterystyczne właściwości omawianego SWD były związane ze strukturą organizacyjną ZPO „Bytom", dla których ten system wykonano. ZPO „Bytom" są dużym przedsiębiorstwem produkującym odzież męską i nastawionym głównie na eksport. Nasze wcześniejsze doświadczenia, zgromadzone przy budowie i zastosowaniu komputerowych modeli wspomagania decyzji kierowniczych ukierunkowały nasze prace na konstrukcję złożonego z wielu modułów SWD finansów. W ten sposób powstał SWD złożony z siedmiu modułów:

PRODUKCJA,
SPRZEDAŻ,
KOSZT WŁASNY,
ULGI,
WYNIK,
ZYSK,
PODZIAŁ.

Główne elementy kształtujące wynik finansowy przedsiębiorstwa, znajdują się w module *ZYSK*. Dla umożliwienia w prowadzenia przez decydenta różnych rozwiązań oraz oceny ich wpływu na zysk, moduł ten rozbudowano o pozostałe sześć modułów. W zależności od potrzeb SWD można wykorzystać w różny sposób. Symulacja może obejmować cały przebieg działalności gospodarczej przedsiębiorstwa (od sprzedaży wyrobów i usług, aż do ustalenia wyniku ekonomicznego oraz jego podziału) lub konkretny wynik działalności gospodarczej (np. koszty sprzedaży lub zysk) widziany oczywiście z perspektywy kierownictwa całego przedsiębiorstwa.

System wspomagania decyzji w obszarze nagradzania (płac)

Zagadnienie wspomagania decyzji w obszarze nagradzania (płac) podzielono wspólnie z użytkownikiem (ZPO „Bytom") na trzy moduły:

ZATRUDNIENIE,
WYNAGRODZENIA,
WYKORZYSTANIE CZASU PRACY.

dokończenie na s. 27

JACEK KOTULA
JAROSŁAW ZYGUŁA
Zakład Elektroniki i Informatyki
Uniwersytet Śląski
Katowice

Model konwersacyjny CAL w środowisku KnowledgePro*

Projektowanie pakietów autorskich ukierunkowanych na tworzenie programów wspomagających nauczanie stanowi interesujące zagadnienie informatyczne [3]. Produkt finalny systemu autorskiego, tj. wygenerowana aplikacja typu CAL (*Computer Aided Learning*) winna być wyposażona w:
- multimedialne, interaktywne środki prezentacji i komunikowania się z użytkownikiem,
- środki umożliwiające reprezentację bazy wiedzy,
- bogaty system konwersacji i wartościowania odpowiedzi,
- automatyczne dostosowanie procesu nauczania do poziomu wiedzy i indywidualnych możliwości uczącego się.

W latach 1987–1992 w Zakładzie Elektroniki i Informatyki Uniwersytetu Śląskiego prowadzono prace nad zagadnieniem szeroko rozumianego oprogramowania dydaktycznego. Po opracowaniu serii specjalizowanych aplikacji typu CAI (*Computer Assisted Instruction*) rozpoczęto prace dotyczące pakietów autorskich. Ich pierwszym efektem było opracowanie systemu autorskiego ASYS 3.0 [1] pracującego pod kontrolą systemu operacyjnego DOS. Pojawienie się systemu Windows 3.0 firmy Microsoft, rozpowszechnienie środków multimedialnych oraz doświadczenia zebrane w toku wcześniejszych prac ukierunkowały dalsze badania w stronę pełniejszego wykorzystania komputerów jako narzędzi dydaktycznych.

We wstępnej fazie przedmiotem szczegółowych badań zespołu był pakiet *AuthorWare Professional for Windows* (APW) firmy Authorware Inc. – profesjonalny pakiet autorski dla środowiska Windows. Podjęto również starania o inne pakiety programowe o podobnych do APW cechach; takich jak SHIVA – produkt projektu DELTA (krajów EWG) oaz Icon Author – oprogramowanie amerykańskiej firmy Digital. Starania naszego Zakładu w kierunku pozyskania tych produktów nie dały pozytywnych rezultatów z uwagi na bariery finansowe i formalne (bardzo wysoki koszt oprogramowania, przepisy celne i inne).

Pakiet APW [5] funkcjonuje na rynku od 1987 r. (na komputerze Apple Macintosh) będąc systemem podlegającym stałemu rozwojowi. Porównując go z materiałami reklamowymi pozostałych pakietów można uznać, że jest on systemem reprezentatywnym dla tej klasy oprogramowania. Spełnia on większość wymagań stawianych pakietom autorskim CAL. Jego podstawowym mankamentem jest jednak znaczna pracochłonność tworzenia aplikacji, nieczytelność struktury przy rozbudowanych lekcjach, praktyczny brak możliwości reprezentacji wiedzy oraz małą podatność na zaimplementowanie w środowisku modeli konwersacyjnych CAL.

Ograniczone możliwości wykorzystania gotowych pakietów autorskich (wymagania stawiane finalnym produktom CAL) spowodowały podjęcie prac nad stworzeniem oryginalnego systemu. Spośród narzędzi programistycznych systemu Windows autorzy wybrali jako rokujący duże nadzieje pakiet KnowledgePro firmy Knowledge Garden Inc. Stwarza on bardzo silne środowisko projektowe, dostarczające narzędzi umożliwiających spełnienie większości wymagań stawianych autorskim systemom projektowania aplikacji CAL.

Pakiet KnowledgePro

KnowledgePro, przeznaczony dla środowiska MS Windows jest sterowanym zdarzeniami obiektowym językiem programowania systemów wykorzystujących bazę wiedzy. Można go zakwalifikować do klasy systemów ekspertowych typu narzędziowego (tzw. SHELL).

Jego podstawowymi własnościami są:
- możliwość definiowania problemów rozwiązania (*Goals*),
- definiowanie faktów i zdarzeń (*Facts*) oraz związanych z nimi reguł wnioskowania (*Rules*),
- zwiększenie efektywności ekspertyzy przez odrzucanie danych i reguł nie mających powiązania z rozwiązywanym problemem (*Pruning*),
- oddzielanie informacji relewantnej od nierelewantnej.

Podstawowe mechanizmy KnowledgePro:
- wewnętrzny system wnioskowania (*Inference Mechanism*),
- możliwości budowy systemu uczącego się (*Intelligent Expert System*) z narzędziami zarządzania i dynamicznie zmieniającą się bazą wiedzy,
- zestaw środków umożliwiających projektowanie dialogowego systemu współpracy z użytkownikiem, w tym m.in. implementacja przedstawionych w modelu odpytywań.

Są to znane w systemie Windows struktury, które umożliwiają:
- wybór *K* z *N* pozycji (*CheckBoxes, ListBoxes*),
- wybór 1 z *N* pozycji (*RadioButtons*),
- edycje tekstu (*EditObjects*),
- inne rodzaje wiązania reakcji użytkownika z generowaniem zdarzeń dla systemu (*Buttons, ScrollBars*),
- możliwość interpretowania danych uzyskanych od użytkownika jako symboli terminalnych języka.

*) Badania realizowane w ramach projektu KBN nr 8-0275-91-01 i dofinansowane z programu badań własnych Uniwersytetu Śląskiego nr 8/IPT/92.

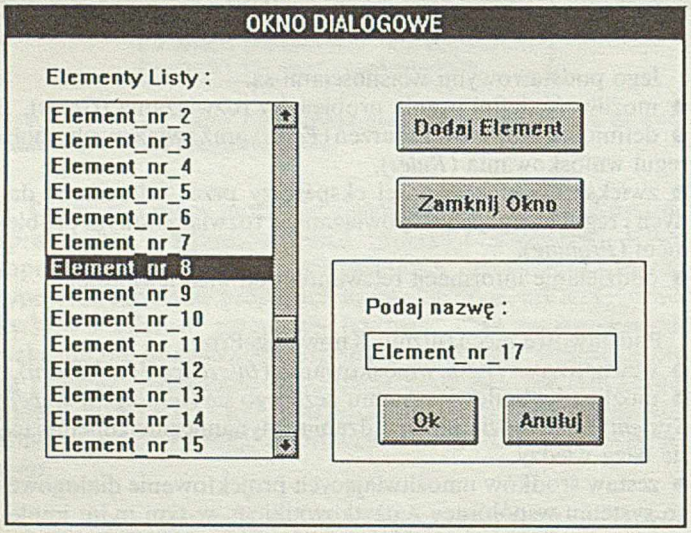
Poza wymienionymi zaletami, pakiet KnowledgePro, jako narzędzie projektowania systemów ekspertowych, dysponuje także typowymi dla języków programowania ogólnego przeznaczenia konstrukcjami sterującymi i modelami danych oraz modułem zarządzania bazami danych w formacie DBase.

Środki manipulowania tekstem oraz strukturami listowymi, a także dostępność operacji logicznych i arytmetycznych, stwarzają przesłanki twierdzenia, że język obiektowy KnowledgePro stwarza możliwości zbudowania efektywnego systemu konwersacji aplikacji CAL (wartościowanie odpowiedzi, nauczanie z bazą wiedzy, elementy ekspertyzy).

Poza wyżej wymienionymi zaletami, pakiet KnowledgePro charakteryzuje się także cechami, zmniejszającymi przydatność systemu jako narzędzia tworzenia CAL. Są to między innymi:
- minimalne możliwości prezentacji i manipulacji obiektami graficznymi,
- stosunkowo niewielkie możliwości przetwarzania, mogące dyskwalifikować KnowledgePro dla niektórych zastosowań.

System KnowledgePro ma możliwość importu i eksportu danych za pomocą dynamicznych struktur przekazywania informacji w środowisku Windows – DDE (*Dynamic Data Exchange*) oraz wykorzystywania procedur zawartych w bibliotekach DLL (*Dynamic Link Library*), co umożliwia rozszerzenie pakietu o wszystkie możliwości środowiska MS Windows i własne, tworzone za pomocą języków ogólnego przeznaczenia, biblioteki.

Tworzenie nowych aplikacji odbywa się w środowisku składającym się z edytora, kompilatora, uruchamiacza (ang. *debbuger*), zarządcy plików, suflera kontekstowego oraz zestawu przydatnych przy pisaniu programów, programów pomocniczych. Wraz z pakietem jest dostarczana obszerna dokumentacja [2] opisująca wszystkie możliwości pakietu, jak również obszerny zestaw programów przykładowych, ułatwiających stawianie pierwszych kroków w systemie.



Rys. 1. Widok okna dialogowego wygenerowanego przez przykładowy program KnowledgePro (patrz wydruk 1)

Podstawowym, występującym w pakiecie elementem jest tzw. *topic*, obiekt języka, będący połączeniem pojęć zmiennej i funkcji z języków typu Pascal czy C. *Topic* może mieć określoną wartość lub jej nie mieć. Jeśli *topic* jest funkcją, to jej wykonanie następuje tylko wtedy, gdy jeszcze nie ma on nadanej wartości. Przy pierwszym wykonaniu *topic* jako funkcji – nadawana jest mu wartość zwracana przez funkcję. Następna jego realizacja może nastąpić dopiero po usunięciu wartości *topic* za pomocą funkcji *reset*. Przedstawiona własność *topic* umożliwia umieszczanie w tekście programu reguł wnioskowania oraz stosowanie do nich zasad rozwiązywania, jakie występują w systemach ekspertowych (ang. *backward chaining*). Język KnowledgePro umożliwia jednocześnie tworzenie nowych linii programu (w tym oczywiście nowych reguł) przez sam program, udostępniając tym samym możliwość samorozbudowy oraz „uczenia się" aplikacji. Przykład programu napisanego w języku KnowledgePro, pokazującego łatwość stworzenia prostej aplikacji w środowisku Windows oraz podstawowe zasady gramatyki tego języka przedstawiono na wydruku 1. Okno dialogowe wygenerowane przez ten program przedstawiono na rysunku 1.

```
Przykład ( ).

 topic Przykład ( ).
  Lib is load_library ( 'UserLib.DLL' ).

  Lista is [ ].
  MAINWIN is window ( , , , 54, 20.5, 'OKNO DIALOGOWE',
                      [ popup, showchildren, visible, titlebar] ).
  Text ( '#x4#y2Elementy Listy :' ).
  LISTBOX is list_box ( ?Lista, , 4, 3.5, 20, 14 ).
  set_list_box ( ?LISTBOX, first (-?Lista ) ).
  button ( 'Dodaj Element', Dodanie, 32, 3.5, 15 ).
  button ( 'Zamknij Okno', Zamknięcie , 32, 6.5, 15 ).
  set_focus ( ?LISTBOX ).

   topic Zamknięcie ( ).
    close_window ( ?MAINWIN ).
   end.

   topic Dodanie ( ).
    reset ( Nowy ).
    user ( ?Lib, AddItemToListBox, [ ?Nowy, ?LISTBOX ] ).
   end.
  end.

   topic Nowy.
     Parametry is window_info ( ?MAINWIN ).
     EDITWIN is window ( , element (?Parametry,1)+26, element (?Parametry,2)+10, 25,
                         8,,[ popup, showchildren, visible, dialogframe ], ?MAINWIN ).
     button ( '&Ok',Wprowadź, 4, 6, 8, , ).
     button ( 'Anuluj',Zamknij, 14, 6, 8 ).
     EDITBOX is edit_box ( , , 3, 3.5, 20, 1.5 ).
     text ( '#y2#x3Podaj nazwę : ' ).
     set_focus ( ?EDITBOX ).
     wait ( ).

     topic Wprowadź.
        Nowy is get_text ( ?EDITBOX ).
        Zamknij ( ).
        continue ( ).
      end.

     topic Zamknij.
       close_window ( ?EDITBOX ).
       close_window ( ?EDITWIN ).
      end.
  end.
```

Wydruk 1

Założenia konwersacyjnego modelu struktury CAL

Przy opracowywaniu modelu struktury jednostek CAL [4] wykorzystano doświadczenia zebrane w trakcie tworzenia systemu autorskiego ASYS oraz w trakcie realizacji projektu TEMPUS JEP 1087 (prace związane z systemem Authorware Professional for Windows). Jednostki CAL będą projektowane na podstawie struktury makroramki, tak jak w wersji dla APW.

Każdej makroramce odpowiada *topic* (obiekt języka KnowledgePro) o unikalnej nazwie klasy *MakroFrame*, umieszczany na liście wykonywanych elementów lekcji. Każdy z obiektów klasy *MakroFrame* zawiera w sobie:
- obiekt klasy *Presentation*, zawierający warstwę prezentacyjną,
- obiekt klasy *„Question*, edytowany przy użyciu konwersacyj-

nych narzędzi systemu projektowania lekcji i zawierający algorytm wartościowania odpowiedzi,
- obiekt klasy *Control*, zawierający zestaw zmiennych oraz opis zdarzeń sterujących procesem uczenia.

Przebiegiem lekcji zarządza moduł arbitra definiowanego w procesie projektowania jednostki CAL. Arbiter na podstawie analizy zmiennych obiektu klasy *Control* poszczególnych makroramek podejmuje decyzje o powtórzeniu odpowiednich fragmentów materiału prezentacyjnego i testowego.

W założeniach projektu szczególną uwagę zwrócono na walory użytkowe (łatwość stosowania) systemu autorskiego oraz wykorzystanie wszystkich możliwości MS Windows, takich jak:
- ujednolicony system obsługi,
- system menu,
- okna dialogowe,
- kontekstowy system podpowiedzi.

Wszystkie opcje systemu winny działać w trybie konwersacyjnym w trakcie projektowania jednostki CAL. Program będzie działał pod kontrolą pakietu KnowledgePro. Z tego wynika, że zarówno projektant, jak i użytkownik jednostki CAL, będzie musiał dysponować autoryzowaną wersją tego pakietu.

W fazie projektowania jednostki lekcyjnej jest generowany tekst programu w języku KnowledgePro, zapisywany do zbioru dyskowego. Projektant lekcji nie musi znać tego języka, a tworzenie projektu polega na wybieraniu odpowiednich opcji z menu.

Środki komunikacji pakietu KnowledgePro z językami ogólnego przeznaczenia

Język KnowledgePro nie wyczerpuje wszystkich możliwości środowiska Windows. Pakiet przystosowano jednak do korzystania z funkcji zawartych w zbiorach bibliotecznych DLL (*Dynamic Link Library*). Na rynku istnieją dwa narzędzia do efektywnego tworzenia takich bibliotek: Turbo Pascal for Windows wersja 1,5 oraz Borland C++ wersja 3.0 – oba produkty firmy Borland International, Inc.

Wraz z pakietem KnowledgePro jest dostarczany „interfejs" umożliwiający dołączanie napisanych w języku C funkcji zewnętrznych. Autorzy artykułu proponują podobny interfejs dla języka Turbo Pascal for Windows.

Zewnętrzne funkcje muszą być wygenerowane do zbioru biblioteki dołączanej dynamicznie (DLL). Przed uruchomieniem funkcji należy załadować bibliotekę do systemu zleceniem:

handl_bib is load_library (nazwa_biblioteki)

Uruchomienie funkcji bibliotecznej następuje przez wydanie zlecenia:

lista_wyników is user (?handl_bib, nazwa_funkcji, lista_parametrów)

Nazwa funkcji napisanej w języku Pascal musi zawierać dyrektywę *export* i być zapowiedziana w sekcji *Exports* biblioteki. W przeciwnym wypadku podczas wywołania wystąpi błąd *I_INVALID_FUNCTION*

Pisząc funkcje należy pamiętać, że musi ona zwrócić do systemu wartość 0 typu *INTEGER*, jeśli jej wykonanie przebiegło pomyślnie lub w przeciwnym wypadku – kod błędu.

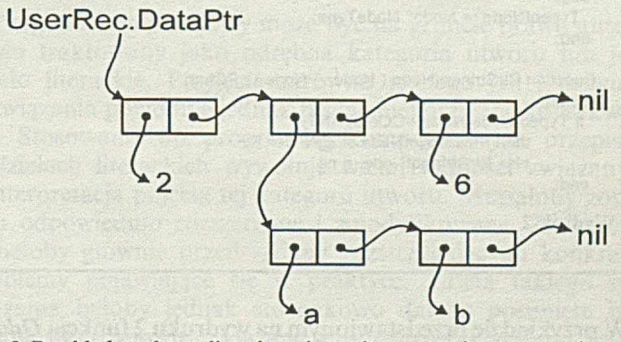
Funkcja przyjmuje wskaźnik do struktury adresowej nazwanej przez producenta oprogramowania *USER_REC*. Struktura ta składa się z sześciu wskaźników, z których trzy są istotne przy pisaniu funkcji zewnętrznych:
- struktury danych importowanych do funkcji,
- struktury danych eksportowanych z funkcji,
- funkcji zaimplementowanej w KnowledgePro, umieszczającej eksportowane struktury na stercie.

Występujące w pakiecie KnowledgePro dane są zapisywane w węzłach danych, powiązanych w listy za pomocą węzłów łączących. Z punktu widzenia programisty piszącego zewnętrzne dla systemu funkcje, istotna jest znajomość struktury następujących trzech podstawowych węzłów:
- numerycznego,
- znakowego,
- łączącego.

Węzeł numeryczny jest rekordem składającym się z dwóch pól:
- identyfikatora węzła – stała typu BYTE o wartości 4,
- zmiennej typu DOUBLE o wartości przechowywanej zmiennej numerycznej.

Węzeł znakowy składa się z trzech pól:
- identyfikatora węzła – stała typu BYTE o wartości 2,
- zmiennej znakowej o dowolnej długości, zakończonej zerem (konwencja C) i przechowującej łańcuch,
- zmiennej typu BYTE o nieistotnej z punktu widzenia programisty wartości.

Zadaniem węzła łączącego jest kojarzenie powyższych węzłów w listy. Składa się on z trzech pól:
- identyfikatora węzła (stała typu BYTE o wartości 1),
- dalekiego wskaźnika, wskazującego na węzeł łączony w listę,
- dalekiego wskaźnika do następnego węzła łączącego.



Rys. 2. Przykład struktury listy danych przekazywanych przy wywoływaniu przez system KnowledgePro procedur zewnętrznych

Sposób tworzenia listy pokazano na rysunku 2. Struktura składa się z trzech pozycji: liczby 2, podlisty o elementach a i b (łańcuchy) oraz liczby 6. Prostokątami zaznaczono węzły łączące. Wskaźniki *nil* oznaczają koniec listy.

Istotnym, występującym przy pisaniu funkcji użytkowania, zagadnieniem jest odpowiednia obsługa sterty programu. W funkcji zewnętrznej należy przydzielać miejsce w pamięci dla eksportowanej listy tak, aby po odczytaniu danych program wywołujący mógł zwolić odpowiednie komórki pamięci. W tym celu zaproponowano trzy funkcje, wykorzystujące wskazanie w rekordzie *USER_REC:*
AllocNodeCon – generowanie węzła łączącego,
AllocNum – generowanie węzła numerycznego,
AllocStr – generowanie węzła znakowego.

Wykorzystując te trzy funkcje można w prosty sposób generować nawet dość rozbudowane listy.

```
unit KPutils;

interface

const
CONSNODE    =  1;
CONSTANT    =  2;
NUMBER      =  4;
NILNODE     =  255;

type
 PNode = ^Node;
 Node  = record
       NodeType : Byte;
       case Byte of
        CONSNODE    : ( Head:       PNode  ;
                        Tail:  PNode  );
        CONSTANT    : ( Str   : Pchar  );
        NUMBER      : ( Number : Double );
       end;

 AllocFunType  =  procedure ( size:longint; var Q:pointer ) ;

 UserRecPtr    =  ^UserRec;
 UserRec       = record
                   DataPtr     : PNode;
                   ReturnPtr   : PNode;
                   UserFunc    : Pointer;
                   AllocFun    : AllocFunType;
                   AttachFun   : Pointer;
                   ValueOfFun: Pointer
                 end;

Function  TypeofNode ( Node : PNode ) : byte;
Function  PtrStringofNode ( Node : PNode ) : PChar;
Function  NumberofNode ( Node : PNode ) : double;
Function  BodyNode ( Node : PNode ) : PNode;
Function  NextItem  ( Node : PNode ) : PNode;
Function  AllocNum ( P : UserRecPtr; Num:double ) : PNode;
Function  AllocStr ( P : UserRecPtr; SendStr :  string ) : PNode;
Function  AllocNodeCon ( P : UserRecPtr; Body, Next :PNode ) : PNode;

 implementation

uses WinProcs, WinTypes;

Function TypeofNode ( Node : PNode ) : byte;
 begin
  TypeofNode := Node^.NodeType;
 end;

Function PtrStringofNode ( Node : PNode ) : PChar;
 begin
  if TypeofNode(Node)=CONSTANT
        then PtrStringofNode := @Node^.Str
        else PtrStringofNode := nil;
 end;
```

Wydruk 2

```
Function NumberofNode ( Node : PNode ) : double;
 begin
  if TypeofNode(Node)=NUMBER
        then NumberofNode := Node^.Number
        else NumberofNode := 0;
 end;

Function BodyNode ( Node : PNode ) : PNode;
 begin
  if TypeofNode(Node)=CONSNODE
        then BodyNode := Node^.Head
        else BodyNode := nil;
 end;

Function NextItem ( Node : PNode ) : PNode;
 begin
  if TypeofNode(Node)=CONSNODE
        then NextItem := Node^.Tail
        else NextItem := nil;
 end;

Function AllocNum ( P : UserRecPtr; Num:double ) : PNode;
 var
   wskNode : PNode;
 begin
  P^.Allocfun(1+sizeof(Num),pointer(wskNode));
  wskNode^.Nodetype:=NUMBER;
  wskNode^.Number:=Num;
  AllocNum:=wskNode;
 end;

Function AllocStr ( P : UserRecPtr; SendStr:string ) :PNode;
 var
   wskNode:PNode;
   Strtemp:string;
   LenStr:word;
 begin
  StrTemp:=SendStr+#0;
  LenStr:=ord(StrTemp[0]);
  P^.Allocfun(1+LenStr+1,pointer(wskNode));
  wskNode^.Nodetype:=CONSTANT;
  move(StrTemp[1],wskNode^.Str,LenStr);
  AllocStr:=wskNode;
 end;

Function AllocNodeCon ( P : UserRecPtr; Body , Next :PNode ) : PNode;
 var
   wskNode:PNode;
 begin
  P^.Allocfun(1+8,pointer(wskNode));
  wskNode^.Nodetype:=CONSNODE;
  wskNode^.Head:=Body;
  wskNode^.Tail:=Next;
  AllocNodeCon:=wskNode;
 end;

begin
end.
```

W przykładzie przedstawionym na wydruku 2 funkcja *OddajListe* przekazuje do programu wywołującego listę o zawartości przedstawionej na rysunku 2.

Zaproponowane przez firmę Knowledge Garden Inc. rozwiązanie kojarzenia danych w listy wprowadza wprawdzie pewną warstwę nadbudowy nad przekazywane dane, lecz w znakomity sposób rozszerza możliwości obsługi struktur drzewiastych.

* * *

Przedstawione w artykule uwagi są wynikiem prac badawczych autorów nad przydatnością różnych środowisk programistycznych do tworzenia aplikacji CAL i systemów autorskich. Opisywany pakiet został autorom udostępniony dzięki uprzejmości Hamburger AusbildungsPartnerschaft. Dalsze prace nad rozbudową środowiska systemu autorskiego do projektowania jednostek CAL z wykorzystaniem pakietu KnowledgePro koncentrują się nad możliwością zaawansowanej edycji obiektów graficznych wchodzących w skład warstwy prezentacyjnej lekcji oraz wykorzystaniem dostępnych w języku KnowledgePro środków sztucznej inteligencji.

LITERATURA

[1] Gaik D., Kotula J., Piecha J.: System Autorski ASYS 3.0 INFORMATYKA nr 5, 1992

[2] KnowledgePro Windows. User Manual, Reference Manual. Knowledge Garden, Inc. 1991

[3] Piecha J.: Podstawy projektowania systemów CAI. INFORMATYKA nr 4, 1991

[4] Piecha J.: Design and evaluation of CAL systems. Interactive Multimedia, SIGMA Press Journals, U. K., June 1992

[5] Piecha J., Przybylski A., Zyguła J.: APW – graficzne narzędzie projektowania systemów autorskich. INFORMATYKA nr 6, 1992

[6] Turbo Pascal for Windows. Windows Programming Guide, Windows Reference Guide. Borland International, 1991.

W każdej chwili można zaprenumerować INFORMATYKĘ oraz kupić zaległe numery z bieżącego roku i lat ubiegłych.

ANDRZEJ MATLAK
Międzyuczelniany Instytut Wynalazczości
i Ochrony Własności Intelektualnej
Kraków

Prawne środki ochrony programów komputerowych – możliwe kierunki zmian w USA

Ochrona programów komputerowych jest w ostatnim czasie zagadnieniem niezwykle żywo dyskutowanym. Program komputerowy bardzo często, bez trudu i praktycznie bez kosztów może zostać skopiowany i następnie wielokrotnie użyty. Często odbywa się to bez poważniejszych przeróbek, a więc z bezsporną szkodą dla jego twórcy.

W obecnej chwili nie istnieje jednomyślność w ustawodawstwach krajowych co do zasad optymalnej ochrony programu komputerowego. Dyskusja koncentrowała się dotąd głównie na prawie autorskim i patentowym, ponieważ te akty prawne gwarantują absolutną i wyłączną ochronę. Z teoretycznego punktu widzenia istnieją jednak i inne sposoby rozwiązania zasygnalizowanego wyżej problemu.

Ciekawym głosem w tych rozważaniach jest raport pod tytułem „Programy komputerowe i własność intelektualna na tle zmian technologicznych"[1], opracowany przez Biuro Oceny Rozwoju Technologicznego (Office of Technology Assessment – OTA), która jest ciałem doradczym i opiniodawczym Kongresu Stanów Zjednoczonych.

Raport ten kończy kilkuletni okres prac prowadzonych przy udziale m.in. Urzędu Prawa Autorskiego (*U.S. Copyright Office*), Urzędu Patentowego i Znaków Towarowych (*U.S. Patent and Trademark Office*), Bibloteki Kongresu, Ministerstwa Energetyki, przedstawicieli nauki i przemysłu Stanów Zjednoczonych. Opracowanie analizuje gwałtowne i kompleksowe technologiczne w dziedzinie oprogramowania komputerowego i ich wpływ na system własności intelektualnej. Praca ta stanowi zbiór wskazówek dla Kongresu USA, jakie powinno się rozważyć w momencie podejmowania decyzji związanych z kształtowaniem prawnego reżimu ochrony programów komputerowych.

W wyniku przeprowadzonych badań dostrzeżono trzy grupy problemów:

- trudności wynikające ze stosowania ochrony prawnoautorskiej oprogramowania komputerowego;
- trudności powstające w momencie objęcia programów komputerowych ochroną wynikającą z przepisów prawa patentowego;
- trudności z jakimi borykają się biblioteczne zbiory danych i użytkownicy informacji cyfrowych.

[1] Finding a Balance: Computer Software, Intellectual Property and the Challenge of Technological Change, S/N 052-003-01278, Washington, 1992

Prawo autorskie

Istnieją różne możliwości ukształtowania regulacji prawnoautorskich odnoszących się do oprogramowania komputerowego. W pierwszej kolejności należy zastanowić się nad rozszerzeniem brzmienia § 102 obowiązującego w USA prawa autorskiego tak, aby objąć wyraźnie jego hipotezę języki komputerowe, algorytmy oraz różnego rodzaju interfejsy. Teoretycznie można rozważyć wyraźne wykluczenie takich elementów oprogramowania z regulacji prawnoautorskiej.

W jednej i drugiej sytuacji należy w miarę precyzyjnie określić zakres przedmiotowy takich pojęć, jak język komputerowy, algorytm, itp, które powinny być dostatecznie zrozumiałe dla osób zainteresowanych. W związku z szybkimi zmianami technologicznymi, stworzenie takich definicji wymaga ścisłej współpracy pracowników z twórcami oprogramowania.

Program komputerowy może być na gruncie prawa autorskiego traktowany jako odrębna kategoria utworu lub jego dzieło literackie. Przyjęcie zarówno jednego jak i drugiego rozwiązania powoduje jednak pewne niekorzystne konsekwencje. Stosowanie do programów komputerowych przepisów o dziełach literackich wywołuje wiele trudności związanych z interpretacją pojęcia tej kategorii utworu. Musiałoby zostać ono odpowiednio rozszerzone i zmodyfikowane. Zadanie to stanęłoby głównie przed sądami rozstrzygającymi konkretne problemy pojawiające się w praktyce. Zaletą takiego rozwiązania byłoby jednak stosunkowo daleko posunięte bezpieczeństwo obrotu, związane z możliwością wykorzystania wykształconego już dotychczas orzecznictwa i literatury w omawianej dziedzinie. Uważa się jednak, że przy tego typu konstrukcji zakres ochrony prawnoautorskiej powinien zostać ograniczony. Proponuje się, aby jako program komputerowy rozumieć tylko kod (tzn. tekst). W zakresie tego pojęcia nie mieściłyby się już np. interfejsy czy rozbudowanie programu, który jest już używany. Dla ostatnio z wymienionych przedmiotów należałoby stworzyć odrębną, szczególną regulację prawną (łac. *sui generis*). Przyjęcie takiego rozwiązania pozwoliłoby na kontynuowanie ochrony kodu programu zarówno w Stanach Zjednoczonych, jak i w innych krajach zgodnie z postanowieniami konwencji berneńskiej.

Konieczne byłoby jednak wówczas stworzenie dodatkowego reżimu ochrony dla innych elementów oprogramowania (takich, jak np. wzory interfejsów użytkownika, zewnętrznej formy programu). Do chwili obecnej nie jest bowiem do końca jasne, w jakim zakresie wymienione elementy mogą być chronione na gruncie prawa autorskiego. Wydaje się więc, iż skonstruowanie dla nich regulacji *sui generis* zmniejszyłoby zdecydowanie

niepewność co do ich ochrony i spowodowałoby rozciągnięcie jej na niechronione dotąd elementy oprogramowania.

Proponowane rozwiązanie mogłoby inaczej określić kryterium „oryginalności", przewidywać krótsze od prawnoautorskich okresy ochrony oraz inaczej statuować sposoby i zakres naruszenia przyznanych praw. Przy tak skonstruowanej regulacji prawnej konieczne byłoby również, w miarę dokładne, rozgraniczenie ochrony oprogramowania komputerowego wynikającego z prawa autorskiego oraz ochrony wynikającej ze wspomnianej regulacji *sui generis*, jak również określenie relacji jakie mogą zachodzić między tymi aktami prawnymi.

W momencie przyznania programom komputerowym ochrony prawnoautorskiej, należy zastanowić się nad możliwością opracowania specjalnego katalogu wskazówek ich używania, zgodnego z zasadami współżycia społecznego (ang. *fair use*). Prace te powinny prowadzić do zmiany § 117 prawa autorskiego, do którego treści należałoby zastosować określenie prawnych granic pojęcia *„fair use"*, odnoszącego się do programów komputerowych, ze szczególnym uwzględnieniem niepublikowanych dzieł oraz problemu dozwolonej dekompilacji.

Jeszcze innym możliwym rozwiązaniem jest całkowite zastąpienie prawnoautorskiej ochrony oprogramowania komputerowego ochroną szczególną (*sui generis*). Regulacja taka obejmowałaby swym zasięgiem nie tylko programy komputerowe ujmowane w całości, lecz również np. interfejsy czy odwrotną analizę inżynieryjną (ang. *reverse engineering*). Rozwiązanie takie mogłoby zawierać wyższy, niż wynikający z prawa autorskiego, standard „oryginalności", przewidywać węższy zakres ochrony oraz jej krótsze terminy. Niewątpliwym mankamentem takiej konstrukcji byłoby wywołanie dużej niepewności obrotu prawnego w dziedzinie oprogramowania. Regulacja taka zrywałaby bowiem z wytworzoną na przestrzeni wielu lat linią orzecznictwa, która powstawała na podstawie licznych, pojawiających się w praktyce stanów faktycznych. Utrudniałaby ona również obrót programami komputerowymi na rynku międzynarodowym, zwłaszcza z państwami przyznającymi omawianym przedmiotom ochronę prawnoautorską.

Prawo patentowe

Inne jakościowo problemy powstają w związku z ochroną programów komputerowych w ramach prawa patentowego. Należy tu w pierwszej kolejności określić zakres zdolności patentowej wynalazków związanych z oprogramowaniem.

Od wczesnych lat sześćdziesiątych zarówno sądy, jak i *U.S. Patent and Trademark Office* starają się określić, czy i do jakiego stopnia programy komputerowe winny korzystać z ochrony patentowej. Do chwili obecnej amerykański Sąd Najwyższy zbadał wiele spraw (m.in. Gottschalk v. Benson, Parker v. Flook, Diamond v. Diehr), próbując ustalić granice zdolności patentowej w omawianej materii.

Na gruncie dominującej obecnie interpretacji przepisów prawa patentowego, patenty mogą być udzielane na pewne części funkcji programów. W zależności od tego, które funkcje programu są nowe, nieoczywiste i spełniają inne ustawowe przesłanki, ten sam program może zawierać w sobie wiele wynalazków posiadających zdolność patentową lub może nie mieć ich wcale.

Aby zredukować niepewność i wyjaśnić sytuację prawną w omawianej dziedzinie, Kongres Stanów Zjednoczonych winien w pierwszej kolejności zmienić ustawową definicję zdolności patentowej w ten sposób, aby precyzyjnie określić, które elementy oprogramowania lub algorytmy znajdują się w jej obrębie.

Na gruncie prawa patentowego niezwykle istotne jest sprawdzenie, czy wynalazek jest nowy i czy nie wynika w sposób oczywisty z istniejącego w danej dziedzinie stanu techniki (ang. *prior art*). Dlatego też ustalenie *prior art* jest pierwszoplanową czynnością w procesie badania zgłoszonego wynalazku. W tym miejscu pojawiają się jednak znaczne trudności. Są one wywołane w pierwszym rzędzie niewystarczającym przygotowaniem pracowników *U.S. Patent and Trademark Office*, zajmujących się badaniem wynalazków związanych z programami komputerowymi. Taka sytuacja spowodowana jest przede wszystkim bardzo szybkim rozwojem technologicznym w dziedzinie oprogramowania. Wskazanym byłoby więc stworzenie z jednej strony systemu ciągłego podnoszenia kwalifikacji osób, zajmujących się oceną zgłaszanych wynalazków (np. przez tworzenie z dodatkowo przyznanych funduszy specjalistycznych form szkolenia), a z drugiej – opracowanie specjalnego katalogu informacji, który umożliwiałby określenie aktualnego stanu techniki w dziedzinie programów komputerowych. Przy tworzeniu tego zbioru powinni brać udział przedstawiciele wszystkich zainteresowanych podmiotów, m.in. *U.S. Patent and Trademark Office*, twórców, producentów i użytkowników oprogramowania. Katalog taki powinien obejmować nie tylko dokładny opis wynalazków opatentowanych, lecz również tych, którym odmówiono ochrony. Powszechność tego typu informacji ograniczałaby niepewność obrotu w omawianej dziedzinie i mogłaby prowadzić do bardziej wszechstronnej oceny zgłaszanych wynalazków.

W związku z szybkimi zmianami technologicznymi w dziedzinie programów komputerowych (powodującymi znaczną dezaktualizację opatentowanych już wynalazków), dane te powinny być publikowane stosunkowo szybko (np. po upływie 18 miesięcy od dnia zgłoszenia). Katalog winien być ponadto uzupełniany w taki sposób, aby stale zachowywał swoją aktualność. Spełniałby on wówczas dwie podstawowe funkcje. Z jednej strony stanowiłby podstawowy zbiór wiadomości dla pracowników *U.S. Patent and Trademark Office*, wykorzystywany w procesie badania zgłaszanych wynalazków, a z drugiej pełniłby szeroko rozumianą funkcję informacyjną w stosunku do wszystkich podmiotów zainteresowanych problematyką komputerową.

Oprócz wyżej omówionych możliwości należałoby rozważyć również sposoby zainteresowania przedmiotową problematyką prawną szerokiego kręgu osób, związanych z tworzeniem i użytkowaniem oprogramowania. Osoby te mogłyby dostarczyć bogatego materiału informacyjnego, potrzebnego do ustalenia zarówno zdolności patentowej poszczególnych kategorii wynalazków, jak i samej procedury przyznawania im ochrony.

Uwzględnienie zasygnalizowanych wyżej rozwiązań powinno wpłynąć na wzrost poziomu badań prowadzonych przez *U.S. Patent and Trademark Office*, decydujących o przyznaniu w konkretnym przypadku ochrony patentowej.

Inną możliwością jest całkowite wykluczenie wynalazków odnoszących się do oprogramowania komputerowego i/lub algorytmów z zakresu prawa patentowego oraz stworzenie dla nich specjalnej ochrony *sui generis*.

Bazy danych

W praktyce wiele problemów dostarcza również ocena prawnego charakteru bibliotecznych baz danych oraz zakresu uprawnień osób korzystających z różnego rodzaju informacji cyfro-

wych. Zwłaszcza wykorzystywanie informacji gromadzonych i udostępnianych przy użyciu technologii elektronicznej wzbudza żywe zainteresowanie wśród szerokiego kręgu podmiotów (w szczególności zakładów przemysłowych, instytucji oraz ogromnej rzeszy prywatnych odbiorców). W związku z tym postuluje się stworzenie w miarę jasnych granic uczciwego korzystania z tej formy informacji (*fair use*).

Przy tworzeniu tego typu zaleceń należy sięgnąć do istniejącego już katalogu zachowań, określającego granice *fair use* w zakresie korzystania z tradycyjnych usług bibliotecznych. Uprawnienia bibliotek odnoszące się do przechowywania w archiwach, wyposażania oraz udostępniania tradycyjnych materiałów wynikają bowiem z przepisów prawa autorskiego.

W przypadku informacji cyfrowych sytuacja wygląda jednak nieco inaczej. Wynika to ze specyfiki tego typu zbiorów, które mogą być znacznie częściej i na dużo szerszą skalę używane przez osoby korzystające z usług bibliotecznych. Dlatego też w odniesieniu do informacji cyfrowych, uprawnienia przysługujące poszczególnym podmiotom są określane w odrębnych umowach z dostawcami tego typu danych.

Istnieje także konieczność stworzenia możliwie szerokiego katalogu wskazówek odnoszących się *fair use* w dziedzinie bibliotecznych baz danych, informacji cyfrowych, jak również dzieł udostępnianych w formie elektronicznej. Wskazówki te powinny w miarę dokładnie określać zakres pojawiających się w tej materii pojęć (takich jak kopiowanie, używanie, czytanie informacji). Mogą one przewidywać również określone ograniczenia praw autorskich z uwagi na ważny interes społeczny (dozwolony użytek publiczny).

W omawianym raporcie postuluje się również opracowanie specjalnego zbioru opłat, należnych właścicielom praw autorskich w związku z udostępnianiem ich dzieł w formie elektronicznej.

* * *

Warto zaznaczyć, iż każdy z rozważanych reżimów ochrony programów komplikuje nie tylko złożoność i różnorodność samego oprogramowania, ale również ogromne tempo zmian w dziedzinie technologii komputerowej. Ustawodawstwo i orzecznictwo często nie są w stanie nadążyć z rozwiązywaniem pojawiających się nowych problemów o charakterze technicznym. W związku z tym postuluje się powołanie specjalnej komisji Kongresu Stanów Zjednoczonych ds Technologii Komputerowej. W ślad takiego organu wchodziliby wybitni specjaliści zajmujący się na co dzień oprogramowaniem, powoływani spośród przedstawicieli nauki oraz przemysłu. Komisja taka przygotowywałaby okresowe raporty na temat kierunków i zakresu zmian pojawiających się w dziedzinie oprogramowania. Raporty te ułatwiłyby Kongresowi wybór odpowiedniej, dostosowanej do realiów technicznych polityki ustawodawczej.

Pewne mankamenty dostrzeżono w procesie funkcjonowania organów wymiaru sprawiedliwości, których kadra nie jest dostatecznie zorientowana w problematyce związanej z oprogramowaniem kmputerowym. Efektem tego są bardzo długo trwające procesy, jak i nie zawsze najtrafniejsze (z merytorycznego punktu widzenia) ich rozstrzygnięcia. Dlatego warto zastanowić się również nad zorganizowaniem specjalistycznych form szkolenia sędziów, którzy w przyszłości mieliby rozstrzygać pojawiające się w praktyce wątpliwości. Wskazane wyżej propozycje mogłyby wpłynąć na wzrost bezpieczeństwa obrotu i zagwarantować twórcom oraz producentom programów komputerowych bardziej efektywną ochronę.

Wydaje się, że wyniki relacjonowanego tu raportu, winny być brane pod uwagę w Polsce, zarówno przy przygotowywaniu przez Urząd Patentowy Kodeksu Ochrony Własności Przemysłowej, jak i w pracach nad nowym prawem autorskim.

Systemy wspomagania decyzji z bazą wiedzy

dokończenie ze s. 20

Parametry charakteryzujące te metody odnoszą się zarówno do całego przedsiębiorstwa, jak i poszczególnych jego jednostek organizacyjnych (zakładów). SWD zbudowano za pomocą generatora *EUREKA*.

W module *ZATRUDNIENIE* znajdują się informacje dotyczące zatrudnienia w poszczególnych jednostkach organizacyjnych ZPO „Bytom". Dane te są podstawą do tworzenia informacji zagregowanych wskaźników oraz analiz struktury zatrudnienia. W module *WYNAGRODZENIA* znajdują się informacje o wynagrodzeniach poszczególnych grup pracowniczych. Moduł ten ułatwia analizę problemów nagradzania pracowników. W module *WYKORZYSTANIE CZASU PRACY* podstawą bazy danych są informacje o czasie pracy, postojach itp. Moduł ten jest ważnym narzędziem wspomagania analizy czasu pracy, struktury czasu pracy, struktury przestojów itp.

LITERATURA

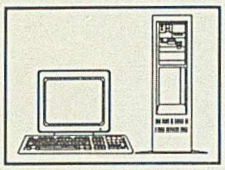
[1] Alter S.: DSS, Addison – Wesley, Massachusetts 1980
[2] Kast F.E., Rosenzweig J.E.: Organization and Management, McGrow-Hill, Tokyo 1982
[3] Klein M., Mathlie L.B.: Expert Systems: A Decision Support Approach, Addison – Wesley 1990
[4] Materiały z konferencji IFIP: WG 8.3 Decision Support Systems w latach 1985–90
[5] Materiały z cyklu konferencji z lat 1980, 1985, 1988, 1990, 1992, obejmujących problematykę SWD. Wyd. AE Katowice
[6] Prace badawcze dotyczące metodologii i konstrukcji SWD koordynowane przez Akademię Ekonomiczną w Katowicach w latach 1984–1990, realizowane w zespołach Z. Hellwiga – AE Wrocław, W. Olejniczaka – Uniwersytet Szczeciński, H. Sroki i St. Stanka AE Katowice
[7] Projekty SWD i wdrożone systemy wspomagania decyzji wykonane w Katedrze Informatyki Akademii Ekonomicznej w Katowicach w latach 1986–1990
[8] Simon H.: Information. Processing Models of Congition. Journal of USA Society for Information Science, 1981
[9] Sroka H.: Systemy wspomagania decyzji kierowniczych. AE Katowice 1986
[10] Sroka H.: Memory Processes and Design and Decision Support. 11–SPUDM Conference, Cambridge 1987
[11] Treauf R.J.: Expert systems in finance and accounting. Greenwood Press, 1990.

Ogłoszenie w INFORMATYCE
dotrze przede wszystkim do profesjonalistów!

Informujemy, że począwszy od marca br. wpłaty na prenumeratę INFORMATYKI są przyjmowane nie tylko przez Zakład Kolportażu Wydawnictwa SIGMA-NOT, lecz także przez wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczycieli na terenie całego kraju.

Kalinowski T.: Programowanie systemów transputerowych INFORMATYKA 1993, nr 4, s. 1 Charakterystyka rozwoju architektury i języków programowania systemów transputerowych oraz metod przygotowania i uruchamiania programów dla tych systemów.	**Kalinowski T.: Transputer system programming** INFORMATYKA 1993, No. 4, p. 1 Characteristics of architecture and programming languages development for transputer systems, as well as of program's preparation and debugging for such systems.	**Kalinowski T.: Programmierung von Transputersystemen** INFORMATYKA 1993, Nr. 4, S. 1 Eine Charakteristik von Architektur- und Programmiersprachenentwicklung der Transputersysteme, sowie von Vorbereitung und Inbetriebsetzung der Programme für solche Systeme.
Korbel L., Miłosz E., Miłosz M.: Systemy zarządzania relacyjnymi bazami danych i ich przydatność do budowy informatycznych systemów zarządzania INFORMATYKA 1993, nr 4, s. 9 Wyniki badań i analiza efektywności działania czterech najczęściej stosowanych systemów zarządzania relacyjnymi bazami danych: PROGRESS, INGRES, ORACLE oraz INFORMIX.	**Korbel L., Miłosz E. Miłosz M.: Relative database management systems and their usebility for building of management information systems** INFORMATYKA 1993, No. 4, p. 9 Research results and analysis of operating effectiveness of the four mostly applied relative database management systems: PROGRESS, INGRES, ORACLE and INFORMIX.	**Korbel L., Miłosz E., Miłosz M.: Relationsdatenbankverwaltungssysteme und ihre Eignung für den Bau von Verwaltungsinformationssystemen** INFORMATYKA 1993, Nr. 4, S. 9 Forschungsergebnisse und Analyse der Betriebsleistung von den vier am meisten angewendeten Relationsdatenbankverwaltungssystemen: PROGRESS, INGRES, ORACLE und INFORMIX.
Szyjewski Z.: Klasyfikacja narzędzi wspomagania CASE INFORMATYKA 1993, nr 4, s. 13 Charakterystyka współczesnych metod klasyfikacji narzędzi projektowania systemów informatycznych typu CASE.	**Szyjewski Z.: Classification of CASE assisting tools** INFORMATYKA 1993, No. 4, p. 13 Characteristics of contemporary methods for classification of the CASE data processing system design tools.	**Szyjewski Z.: Klassifizierung von CASE-Hilfsmitteln** INFORMATYKA 1993, Nr. 4, S. 13 Eine Charakteristik von zeitgenössischen Klassifizierungsmethoden der CASE-Hilfsmittel für computerunterstützte Projektierung der Datenverarbeitungssysteme.
Sroka H.: Systemy wspomagania decyzji z bazą wiedzy INFORMATYKA 1993, nr 4, s. 16 Koncepcja systemów wspomagania decyzji z bazą wiedzy oraz przykłady jej realizacji.	**Sroka H.: Systems for decision assisting with knowledge base** INFORMATYKA 1993, No. 4, p. 16 The idea of systems for decision assisting with knowledge base and examples of its realization.	**Sroka H.: Entscheidungsunterstützungssysteme mit einer Wissenbank** INFORMATYKA 1993, Nr. 4, S. 16 Eine Konzeption von Entscheidungsunterstützungssystemen mit einer Wissenbank und Beispiele ihrer Realisierung.
Kotula J., Zyguła J.: Model konwersacyjny CAL w środowisku KnowledgePro INFORMATYKA 1993, nr 4, s. 21 Charakterystyka pakietu KnowledgePro jako narzędzia do projektowania autorskich systemów wspomagających nauczanie typu CAL.	**Kotula J., Zyguła J.: The CAL conversation model in the KnowledgePro environment** INFORMATYKA 1993, No. 4, p. 21 Characteristics of the KnowledgePro program package as a tool for designing of CAL – computer assisted teaching systems.	**Kotula J., Zyguła J.: CAL-Konversationsmodell in der Umwelt von KnowledgePro** INFORMATYKA 1993, Nr. 4, S. 21 Eine Charakteristik von dem Knowledge Pro-Softwarepaket als Hilfsmittel zur Projektierung der CAL-Lehrunterstützungssysteme.
Matlak A.: Prawne środki ochrony programów komputerowych – możliwe kierunki zmian w USA INFORMATYKA 1993, nr 4, s. 25 Omówienie aktualnego amerykańskiego raportu na temat prawnej ochrony programów komputerowych uwzględniającego wpływ zmian technologii informatycznej.	**Matlak A.: Legal media for computer programs protection – possible directions of changes in the USA** INFORMATYKA 1993, No. 4, p. 25 Discussion of the actual american report on legal protection of computer programs, which includes the influence of information technology changes.	**Matlak A.: Gesetzmittel für Computerprogrammeschutz – mögliche Veränderrungsrichtungen in den Vereinigten Staaten** INFORMATYKA 1993, Nr. 4, S. 25 Eine Besprechung von aktuellen amerikanischen Bericht über den Gesetzschutz der Computerprogramme, der den Einfluss von Veränderungen der Informationstechnologie berücksichtigt.

JUNISOFTEX®



Przedsiębiorstwo Produkcyjno-Usługowe Innowacji Technicznych Biuro Techniczno-Handlowe 44-100 Gliwice ul. Konstytucj 11.
tel. 31-57-00, 31-51-31, 31-75-10, fax 31-75-10, centr. 31-90-81 do 88, telex 036233
Zespół wdrożeń i analiz systemów komputerowych wewn. 15 ,25, 282, Zespół inżynierii oprogramowania wewn. 14, 20, 245,
Zespół d/s serwisu systemów Novell i sprzętu komputerowego wewn. 16, 245, Dział Handlowy wewn. 18, 28, 201,
Szkoła Komputerowa wewn. 282, Sekretariat wewn. 250, Prezes wewn. 13, 272, Dyrektor wewn. 12, 272, Główny Księgowy wewn. 256,
** Konto: PKO I/O Gliwice 27661-225298-136 **

OFERUJEMY

Kompleksową komputeryzację przedsiębiorstw obejmująca dostawę i wdrożenie zintegrowanych systemów komputerowych takich jak system:

finansowo-kosztowy
(automatyczne rozliczanie kosztów)
+
gospodarka materiałowa
+
środki trwałe
+
wyposażenie
+
obrót towarowy fakturowanie sprzedaży
+
kadry-płace
+
techniczne przygotowanie produkcji

Sprzęt komputerowy w tym:

Komputery
Monitory
Dyski twarde
Drukarki
(mozaikowe i laserowe)
osprzęt sieciowy
(karty sieciowe, kable zwykłe i światłowodowe)

Systemy i oprogramowanie licencjonowane

DOS
Novell
Btrieve
SQL Base
Windows

Szkolenia komputerowe

podstawy obsługi komputerów
arkusze kalkulacyjne (Qpro, Lotus, EXCEL)
edytory tekstów (WordPerfect, TAG, ChiWriter)
bazy danych (dbase IV, Clipper, Paradox)
systemy operacyjne (DOS, Novell)

SOFTWARE SUPPORT FOR DOS, NOVELL 2.2, 3.11, 4.0, SQL BASE firmy GUPTA

*

OPROGRAMOWANIE DLA PRZEDSIĘBIORSTW

oferujemy własnej produkcji zakładowy system informatyczny PHU-Perfect(N) obejmujący swoim zakresem:
system finansowo-kosztowy umożliwiający automatyczne rozliczanie kosztów, atomatyczną dekretację, analizę rozrachunków
system obrotu materiałami, wyrobami, towarami (kody paskowe) w tym: stany magazynowe, ewidencje zamówień, fakturowanie sprzedaży (VAT) z automatyczną dekretacją rozdzielników kosztów i sprzedaży do systemu finansowo-kosztowego,
system ewidencji majątku trwałego i wyposażenia
system kadrowo-płacowy (SQL) z automatyczną dekretacją rozdzielników płacowych do systemu finansowo-kosztowego
system technicznego przygotowania produkcji (uk. 1993 r)
(oprogramowanie w systemie SQL BASE w przygotowaniu)

*

OPROGRAMOWANIE DLA URZĘDÓW MIAST I GMIN

oferujemy własnej produkcji system finansowo-podatkowy UMG-Perfect(N) obejmujący swoim zakresem:
system finansowo-księgowy,
systemy podatkowe obejmujące: ustalanie wymiarów, naliczanie i kontrolę płatności należności podatkowych dla dowolnych podatników i/lub płatników podatków na ich kontach i kontach dochodów ze swobodną korektą dowolnych informacji w dowolnym okresie a w tym:
ewidencja zapłat z wykorzystaniem kodów paskowych, emisja wezwań do zapłaty, naliczanie odsetek.
system ewidencji majątku

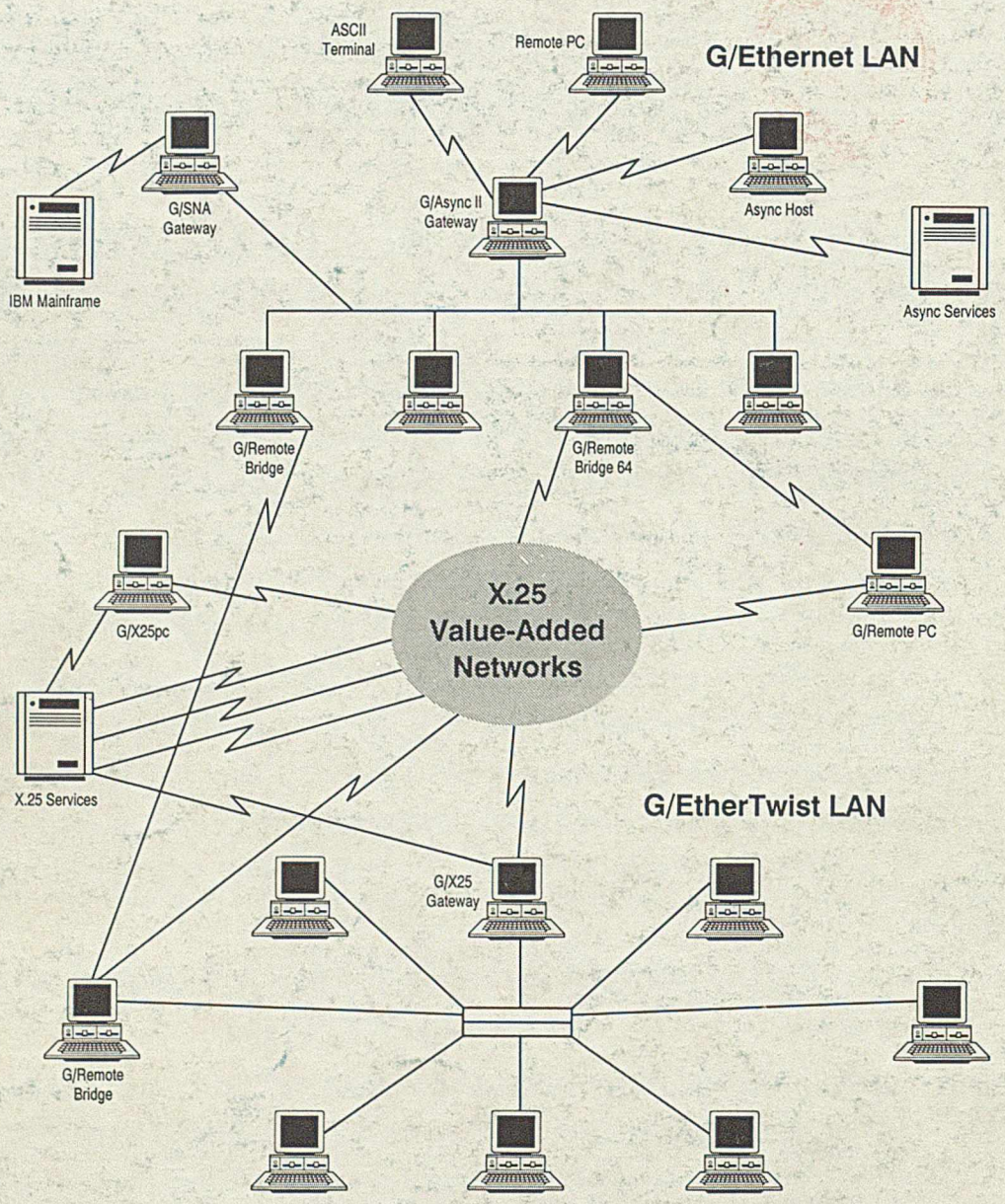
*

Zapraszamy do współpracy w zakresie wdrożeń naszych systemów partnerów, - firmy informatyczne z całego kraju !!!

0/7/92

Produkty sieciowe LAN i WAN

Gateway
communications, inc.



BEZKONKURENCYJNE PRODUKTY
SIECI KOMPUTEROWYCH NAGRADZANE PRZEZ:

PC MAGAZINE, LAN MAGAZINE, INFO WORLD

OFERUJE AUTORYZOWANY DYSTRYBUTOR:

MIKROB

PC MAGAZINE EDITORS' CHOICE January 31, 1989 G/Ethernet AT

PC MAGAZINE EDITORS' CHOICE October 16, 1990 G/EtherTwist

P.W.P.T. MIKROB SP. Z O.O.

20-346 Lublin, ul. Długa 5
Tel. (0-81) 420-61, faks 415-43, teleks 643776 MIKRO PL

0/15/91

Na życzenie wysyłamy katalog produktów. Atrakcyjny program współpracy dla dealer'ów.